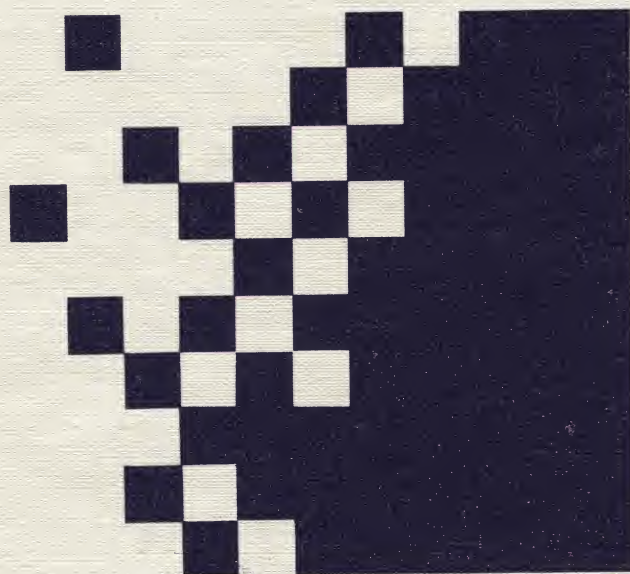


# Operating System

**AST**  
RESEARCH INC.



—AST—  
*Premium*  
COMPUTER PRODUCTS

*Computer Products Library*



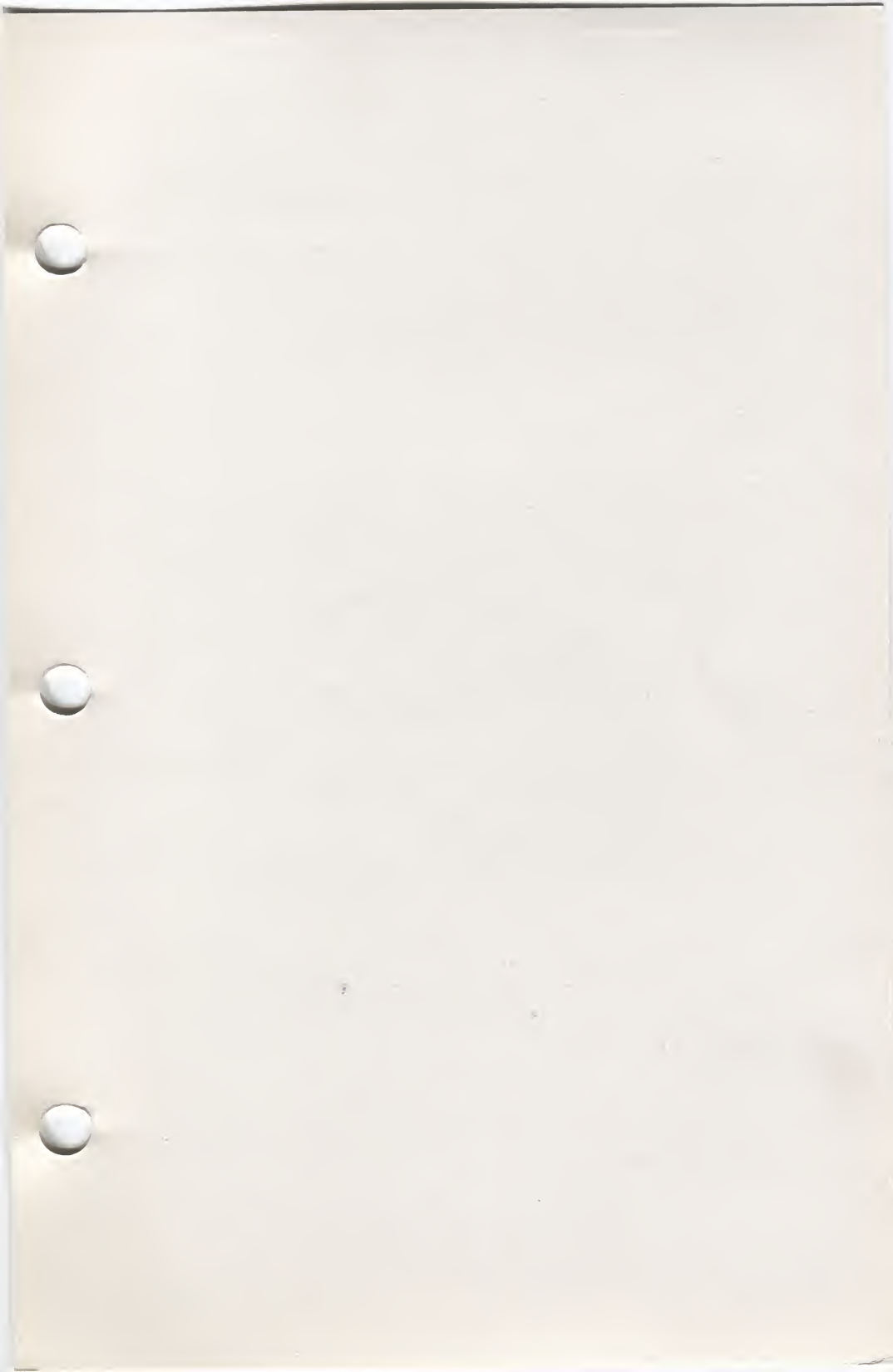


# MS-DOS

**AST**  
RESEARCH INC.



—AST—  
*Premium*  
COMPUTER PRODUCTS





**AST Premium/286™**

**Microsoft®  
Disk Operating System  
(MS-DOS)**

User's Manual  
000399-001 B  
May 1987

AST Research Inc  
Irvine, California  
(714) 863-1333

Second edition (May 1987)

IBM is a registered trademark of International Business Machines Corporation. MS, MS-DOS, XENIX, GW-BASIC, and Microsoft are registered trademarks of Microsoft. Select utilities copyright Phoenix Software Associates, Inc. Wordstar is a trademark of MicroPro.

Changes are periodically made to the information contained in this manual; these changes will be incorporated into new editions.

A product comment form is provided at the back of this publication. If this form has been removed, please address your comments to: AST Research Inc., Attn: Product Marketing, 2121 Alton Ave., Irvine, CA 92714. AST Research may use or distribute any of the information you supply in any form it deems appropriate without incurring any obligations whatsoever.

Copyright © 1987 AST Research, Inc. All rights are reserved, including those to reproduce this book or parts thereof in any form without permission in writing from AST Research, Inc.

### **WARNING**

This manual and software are both protected by United States Copyright law (Title 17 United States Code). Unauthorized reproduction and/or sales may result in imprisonment of up to one year and fines of up to \$10,000 (17 USC 506). Copyright infringers may be subject to civil liability.

# CONTENTS

---

1. FILES AND DIRECTORIES .....	1-1
1.1 Files.....	1-1
1.1.1 What Is a File? .....	1-2
1.1.2 How MS-DOS Keeps Track of Your Files .....	1-3
1.1.3 The DIR (Show Directory) Command .....	1-4
1.1.4 The CHKDSK (Check Disk) Command.....	1-5
1.2 How to Name Your Files.....	1-6
1.3 Invalid File Names.....	1-8
1.4 Wildcards.....	1-9
1.4.1 The ? Wildcard .....	1-9
1.4.2 The * Wildcard .....	1-10
1.5 How To Protect Your Files.....	1-11
1.6 Directories .....	1-12
1.7 Paths.....	1-18
1.7.1 Path Format.....	1-18
1.7.2 Paths and External Commands.....	1-21
1.7.3 Paths and Internal Commands.....	1-22



# CONTENTS

---

1.8 Using Directories.....	1-23
1.8.1 How to Display Your Working Directory .....	1-23
1.8.2 How to Create a Directory .....	1-25
1.8.3 How to Change Your Working Directory .....	1-25
1.8.4 How to Delete a Directory .....	1-26
1.9 How To Rename A Directory .....	1-27
1.10 Summary of Commands.....	1-28
<b>2. LEARNING ABOUT COMMANDS.....</b>	<b>2-1</b>
2.1 What Is a Command? .....	2-2
2.2 Types Of MS-DOS Commands .....	2-3
2.2.1 Internal Commands .....	2-3
2.2.2 External Commands .....	2-4
2.3 Command Options.....	2-5
2.4 Things You Should Know About Commands .....	2-7
2.5 Input and Output .....	2-9
2.5.1 How to Redirect Your Output.....	2-9
2.5.2 Filters.....	2-11
2.5.3 Command Piping.....	2-11
2.6 Batch Processing .....	2-13
2.7 The AUTOEXEC.BAT File.....	2-16
2.8 How To Create A Batch File With Replaceable Parameters .....	2-19
2.9 Command Summary.....	2-22

<b>3. MS-DOS COMMANDS .....</b>	<b>3-1</b>
3.1 About Commands.....	3-1
3.2 About This Chapter .....	3-3
3.3 MS-DOS Commands .....	3-4
ASSIGN .....	3-8
ATTRIB.....	3-10
BACKUP .....	3-12
BREAK.....	3-15
CHDIR.....	3-16
CHKDSK.....	3-18
CLS (Clear screen).....	3-25
COMMAND.....	3-26
COMP (Compare) .....	3-28
COPY .....	3-32
CTTY (Change console) .....	3-40
DATE.....	3-41
DEL (Delete).....	3-43
DIR (Directory) .....	3-44
DISKCOMP (Disk compare) .....	3-46
DISKCOPY .....	3-52
EXE2BIN (Executable to binary) .....	3-58
EXIT .....	3-61
FDISK .....	3-62
FIND .....	3-75
FORMAT .....	3-78
GRAFTABL (Graphics table).....	3-82
GRAPHICS .....	3-83
JOIN .....	3-85
KEYBxx.....	3-87
LABEL.....	3-89
MKDIR (Make directory) .....	3-91

# CONTENTS

---

MODE .....	3-93
MORE .....	3-98
PATH .....	3-99
PRINT .....	3-101
PROMPT .....	3-105
RECOVER.....	3-108
REN (Rename) .....	3-110
REPLACE .....	3-112
RESTORE .....	3-114
RMDIR (Remove Directory) .....	3-116
SELECT .....	3-117
SET .....	3-119
SHARE.....	3-121
SORT .....	3-123
SUBST .....	3-125
SYS (System) .....	3-127
TIME .....	3-129
TREE.....	3-131
TYPE.....	3-133
VER (Version).....	3-134
VERIFY .....	3-135
VOL (Volume).....	3-136
XCOPY .....	3-137
 3.4 Batch Processing Commands.....	3-141
ECHO .....	3-142
FOR .....	3-143
GOTO .....	3-145
IF .....	3-146
PAUSE .....	3-148
REM .....	3-149
SHIFT.....	3-150



4. MS-DOS EDITING AND FUNCTION COMMANDS .....	4-1
4.1 Special Editing Keys .....	4-1
4.2 Control Character Functions .....	4-8
5. EDLIN .....	5-1
5.1 About EDLIN.....	5-1
5.1.1 How EDLIN Works .....	5-1
5.1.2 How to Start EDLIN.....	5-2
5.1.3 How to Quit EDLIN .....	5-3
5.2 Special EDLIN Commands .....	5-4
< F1 > .....	5-5
< F2 > .....	5-6
< F3 > .....	5-7
< Del > .....	5-8
< F4 > .....	5-9
< Esc > .....	5-10
< Ins > .....	5-11
< F5 > .....	5-13
5.3 EDLIN Commands .....	5-15
A (Append) .....	5-16
C (Copy) .....	5-17
D (Delete) .....	5-20
EDIT .....	5-24
E (End) .....	5-26
I (Insert) .....	5-27
L (List) .....	5-32
M (Move) .....	5-35
P (Page) .....	5-38
Q (Quit) .....	5-39
R (Replace) .....	5-40
S (Search) .....	5-44
T (Transfer) .....	5-47
W (Write) .....	5-48

# CONTENTS

---

6. LINK .....	6-1
6.1 Starting and Using LINK .....	6-2
6.1.1 Using Prompts to Specify LINK Files .....	6-2
6.1.2 Using a Command Line to Specify LINK Files .....	6-6
6.1.3 Using a Response File to Specify LINK Files .....	6-9
6.1.4 Using Search Paths with Libraries .....	6-11
6.1.5 The Map File.....	6-13
6.1.6 The Temporary Disk File.....	6-15
6.2 The LINK Options.....	6-16
6.2.1 Viewing the Options List .....	6-18
6.2.2 Pausing to Change Disks .....	6-19
6.2.3 Packing Executable Files.....	6-21
6.2.4 Producing a Public-Symbol Map .....	6-22
6.2.5 Copying Line Numbers to the Map File .....	6-23
6.2.6 Preserving Lowercase .....	6-24
6.2.7 Ignoring Default Libraries .....	6-25
6.2.8 Setting The Stack Size.....	6-26
6.2.9 Setting the Maximum Allocation Space.....	6-27
6.2.10 Setting a High Start Address .....	6-29
6.2.11 Allocating a Data Group .....	6-30
6.2.12 Removing Groups from a Program.....	6-31
6.2.13 Setting the Overlay Interrupt .....	6-32
6.2.14 Setting the Maximum Number of Segments.....	6-33
6.2.15 Using DOS Segment Order .....	6-34
6.3 How LINK Works.....	6-35
6.3.1 Alignment of Segments .....	6-35
6.3.2 Frame Number .....	6-36
6.3.3 Order Segments.....	6-36
6.3.4 Combined Segments .....	6-36
6.3.5 Groups.....	6-38
6.3.6 Fixups .....	6-38

# CONTENTS

---

7. DEBUG.....	7-1
7.1 How to Start DEBUG.....	7-2
7.1.1 DEBUG.....	7-2
7.1.2 Command Line.....	7-3
7.2 DEBUG Commands and Parameters.....	7-4
ASSEMBLE.....	7-8
COMPARE.....	7-11
DUMP.....	7-12
ENTER.....	7-14
FILL.....	7-16
GO.....	7-17
HEX.....	7-19
INPUT.....	7-20
LOAD.....	7-21
MOVE.....	7-23
NAME.....	7-24
OUTPUT.....	7-27
QUIT.....	7-28
REGISTER.....	7-29
SEARCH.....	7-33
TRACE.....	7-34
UNASSEMBLE.....	7-36
WRITE.....	7-38
7.3 DEBUG Error Messages.....	7-40
8. CONFIG.SYS.....	8-1
Index.....	Index-1



# CONTENTS

---

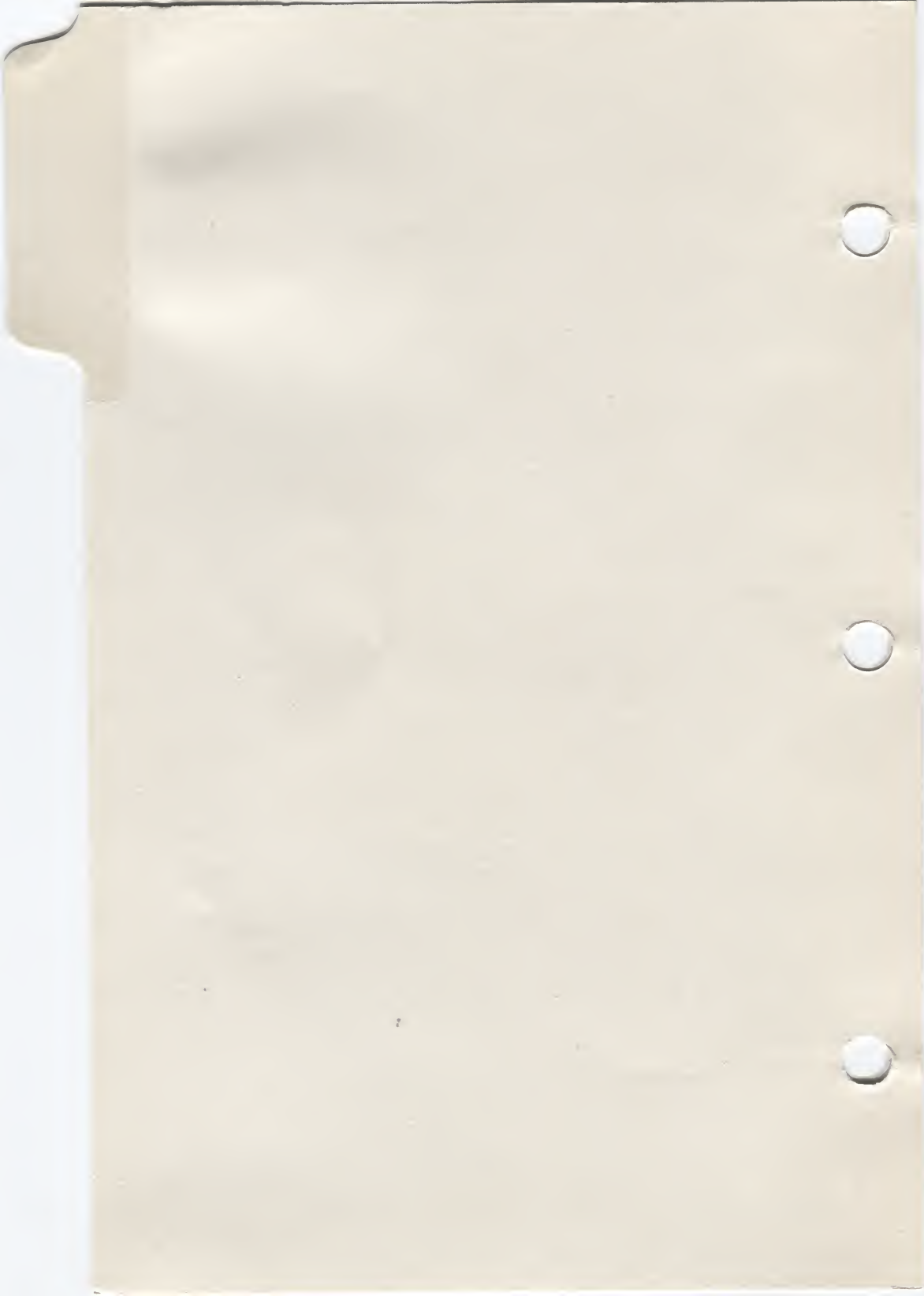
## FIGURES

Figure 1-1. Disk File are like Paper Files.....	1-2
Figure 1-2. Directory and File Allocation Table .....	1-3
Figure 1-3. Multilevel Directory Structure. ....	1-13
Figure 1-4. A Sample Multilevel Directory Structure. ....	1-15
Figure 2-1. How MS-DOS Uses the AUTOEXEC.BAT File .....	2-17
Figure 3-1. FDISK Option Menu .....	3-64
Figure 3-2. Creating a DOS Partition .....	3-65
Figure 3-3. Changing the Active Partition .....	3-69
Figure 3-4. Deleting a DOS Partition.....	3-71
Figure 3-5. Displaying Partition Information .....	3-73
Figure 4-1. Command Line and Template .....	4-2

## Tables

Table 4-1. Special Editing Keys.....	4-3
Table 4-2. Control Character Functions.....	4-8
Table 5-1. Special EDLIN Keys.....	5-4
Table 5-2. EDLIN Commands.....	5-13
Table 6-1. LINK Options .....	6-16
Table 7-1. DEBUG Commands. ....	7-4
Table 7-2. DEBUG Parameters.....	7-6
Table 7-3. DEBUG Flag Codes.....	7-30







This section explains how to use files and directories. Before you read this section, you should already know how to start MS-DOS<sup>®</sup>, format and make backup copies of disks, copy and delete files, and run programs. If you are unfamiliar with any of these tasks, refer to your *AST Premium/286 User's Manual*.

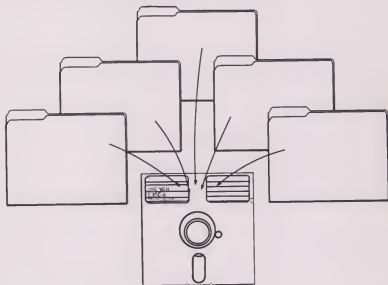
## 1.1 Files

This section covers the following topics:

- What is a file?
- How MS-DOS Keeps Track of Your Files
- The DIR (Directory) Command
- The CHKDSK (Check Disk) Command

### 1.1.1 What is a File?

A file is a collection of related information. A file on your disk can be compared to a file folder in a desk drawer. For example, file folders might contain business letters from clients and important memos from associates. Files on your disks could also contain business letters and memos.



**Figure 1-1. Disk Files are Like Paper Files.**

You create a file each time you enter and save data or text at your terminal. Files are also created when you write programs and save them on your disks. Each file has a unique name. You refer to files by name. In this section, you will learn how to name your files.

### 1.1.2 How MS-DOS Keeps Track of Your Files

The names of files are kept in directories on a disk. Directories also contain the size of the files and the dates they were created and updated. The directory you are working in is called your working directory.

An additional system area, the file allocation table (FAT), tracks the location of your files on the disk. It also allocates the free space on your disks, so that you can create new files.

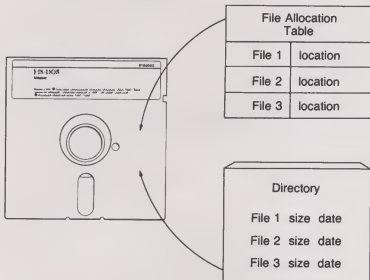


Figure 1-2. Directory and File Allocation Table.

These two system areas, the directories and the FAT, enable MS-DOS to recognize and organize the files on your disks. The FAT is copied onto a new disk when you format it with the MS-DOS Format command, and one empty directory is created, called the root directory.

### 1.1.3 The DIR (Directory) Command

If you want to know what files are on your disk, you can issue the DIR command. DIR tells MS-DOS to display all the files in the working directory on a specific disk. For example, if your MS-DOS disk is in drive A and you want to see the listing for the working directory on that disk, type:

```
DIR A: <Enter>
```

MS-DOS responds with a directory listing of all the files in the working directory on your MS-DOS disk. Each file will display on a separate line, similar to this:

```
COMMAND COM 23769 12-02-85 8:32a
```

where:

*command* is the file name.

*com* is the file name extension.

*12-02-87* is the date the was created, or the most recent date the file was modified.

*8:32a* is the time the file was created or modified.  
*a* indicates a.m., *p* indicates p.m..

After all the files have been listed, the total number of files and the number of bytes free will display.

## NOTE

Two MS-DOS system files, IO.SYS and MSDOS.SYS, are hidden files and will not appear when you issue the DIR command.

You can also get information about any file on your disk by typing DIR and a file name. For example, if you have created a file named MYFILE.TXT, the command:

```
DIR MYFILE.TXT <Enter>
```

gives you a display of all the directory information (name of file, size of file, date created or edited) for MYFILE.TXT.

For more information on the DIR command, refer to Section 3.3.

### 1.1.4 The CHKDSK (Check Disk) Command

The CHKDSK command checks your disks for consistency and errors, much like a secretary proofreading a letter. CHKDSK analyzes the directories and the FAT on the disk you specify. Then CHKDSK produces a status report of any problems, such as files with a non-zero size in the directory that do not contain data.

To check the disk in drive A, type:

```
CHKDSK A: <Enter>
```

MS-DOS displays a status report and any errors it has found. An example of this display and more information on CHKDSK can be found in the description of the CHKDSK command in Section 3.3. You should run CHKDSK occasionally for each disk to make sure the files on your disk are OK.

## 1.2 How to Name Your Files

The name of a typical MS-DOS file looks like this:

NEWFILE.DOC

(filename.extension)

The name of a file consists of two parts. The file name is NEWFILE and the file name extension is .DOC.

A file name can be from 1 to 8 characters long. The file name extension is optional, and can be three or fewer characters. The extension must be separated from the file name by a period.

You can type any file name in small or capital letters. MS-DOS translates letters into uppercase characters. Examples of file names are:

ACCOUNTS.FEB  
BUDGET.84  
SMITHCO.LTR  
CHAPTER1.NVL  
SCHEDULE

In addition to typing the file name and the file name extension, you may need to include a drive name. A drive name tells MS-DOS to look on the disk in a specific drive to find the file name typed. For example, to find directory information about the file NEWFILE.DOC, located on the disk in drive B (when drive B is *not* the current default drive), type the following command:

```
DIR B:NEWFILE.DOC <Enter>
```

Directory Information (name, size, date, and time created) about the file NEWFILE.DOC are displayed on your screen.

If drive A is the default drive, MS-DOS automatically searches the disk in drive A: for the file name NEWFILE, so you don't need to type the drive name. A drive name is needed if you want to tell MS-DOS to look on a different drive to find a file.

Your file names will probably consist of letters and numbers, but other characters are allowed. Valid characters for file name extensions are the same as those for file names. Here is a complete list of the letters and symbols you can use in file names and extensions:

```
A-Z a-z 0-9 $ &
% ' ( ) - @
^ { } ~ \ ! #
```



### 1.3 Invalid File Names

MS-DOS treats some device names differently. Certain three-letter names are reserved for the names of these devices. These three-letter names cannot be used as file names, but they can be used as extensions. You must not name your files any of the following:

AUX	Refers to input from, or output to, an auxiliary device (such as a printer or a disk drive).
CON	Refers to keyboard input, or to output to the terminal console (screen).
PRN	Refers to the printer device.
NUL	Used when you do not want to create a particular file, but the command requires an input or output file name.

Even if you add device designations or file name extensions to these file names, MS-DOS still assumes they refer to the devices listed above. For example, CON.XXX still refers to the console and cannot be used as the name of a disk file.

## 1.4 Wildcards

You can specify two special characters, called wildcards, when searching for files on a disk: the asterisk (\*) and the question mark (?). Wildcards give you greater flexibility when specifying file names in MS-DOS commands.

### 1.4.1 The ? Wildcard

A question mark (?) in a file name or file name extension means that any character can occupy that position. For example, the MS-DOS command:

```
DIR TEST?RUN.EXE <Enter>
```

lists all directory entries on the default drive that begin with TEST, have any character in the next position, end with the letters RUN, and have a file name extension of .EXE.

Here are some examples of files that might be listed by the above DIR command:

```
TEST1RUN.EXE  
TEST2RUN.EXE  
TESTBRUN.EXE
```

### 1.4.2 The \* Wildcard

An asterisk (\*) means that any character can occupy that position or any of the remaining positions in the file name or extension. For example,

```
DIR TEST*.EXE <Enter>
```

Lists all directory entries on the default drive with file names that begin with the characters TEST and have an extension of .EXE. Here are some examples of files that might be listed by the above DIR command:

```
TEST1.EXE  
TEST2RUN.EXE  
TEST6RUN.EXE  
TESTALL.EXE
```

#### NOTE

The wildcard designation \*.\* refers to all files on the disk. This can be very powerful and destructive when used in MS-DOS commands. For example, the command:

```
DEL *.* <Enter>
```

deletes all files on the default drive, regardless of file name or extension.

**Examples:**

To list the for all files named ACCOUNTS on drive A: regardless of their file name extensions, type:

```
DIR A:ACCOUNTS.* <Enter>
```

To list the directory entries for all files with file name extensions of .TXT (regardless of their file names) on the disk in drive B, type:

```
DIR B:*.*.TXT <Enter>
```

This command is useful if, for example, you have given all your text files a file name extension of .TXT. By using the DIR command with wildcard characters, you can get a listing of all your text files even if you do not remember all their file names.

## **1.5 How To Protect Your Files**

MS-DOS is a powerful and useful tool when you process personal and business information. As with any computer, errors can occur and information is sometimes misused.

If your work cannot be replaced or requires a high level of security, make sure your programs are protected from others using, modifying, or even deleting them. Simple actions -- such as removing your disks when they are not in use, keeping backup copies of valuable information, and installing your equipment in a secure facility -- can help you keep your files secure.

## 1.6 Directories

The names of your files are kept in a directory on each disk. The directory includes the size of the files, and the dates they were created and updated.

When more than one person uses your computer, or when you are working on several different projects, the number of files in the directory can become large and unwieldy. You may want your own files kept separate from a coworker's, or you may want to organize your programs into convenient categories.

In an office, you can separate files by putting them in different filing cabinets; in effect, creating different directories of information. MS-DOS lets you organize the files on your disks into directories. Directories are a way of dividing your files into convenient groups of files. For example, you may want all of your accounting programs in one directory and text files in another.

Any one directory can contain any reasonable number of files, and it can also contain other directories (referred to as subdirectories).

This method of organizing your files is called a multilevel directory structure.

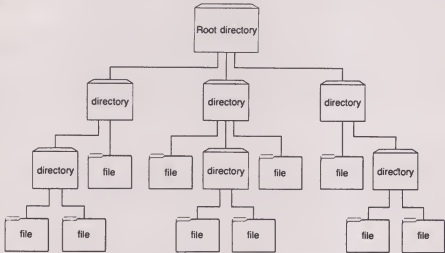


Figure 1-3. Multilevel Directory Structure.

A multilevel directory structure can be thought of as a *tree* structure: directories are branches of the *tree* and files are the leaves, except that the *tree* grows downward. The *root* is at the top. The *root* is the first level in the directory structure. The *root* directory is automatically created when you format a disk and start putting files on it. You can create more directories and subdirectories by following the instructions later in this section.

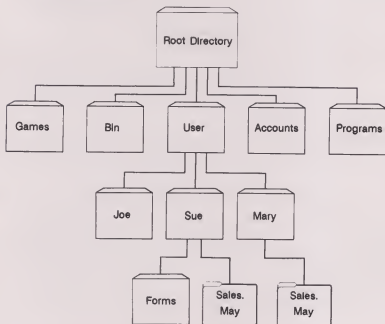
The directory structure grows as you create new directories for groups of files, or for other people on the system. Within each new directory, you can add new files, or create new subdirectories.

You can travel around this structure to find any file in the system by starting at the root, then traveling down any of the branches to the desired file. Conversely, you can start within the file system and travel toward the root.

The directory you are in is called the *working directory*. The file names discussed earlier in this chapter are relative to your working directory, and do not apply to any other directories in the structure. Thus, when you turn on your computer, you are *in* the working directory.



Unless you take special action when you create a file, the new file is created in the working directory. You can have files of the same name that are unrelated because each is in a different directory. See Figure 1-4, which illustrates a typical multilevel directory structure.



**Figure 1-4. A Sample Multilevel Directory Structure.**

The root directory is the first level in the directory structure. You can create subdirectories from the root by issuing the MKDIR command (refer to Section 3.3 for information on MKDIR).

In this example, five subdirectories of ROOT have been created. These include:

1. A directory of games, named GAMES.
2. A directory of all external commands, named BIN.
3. A USER directory containing separate subdirectories for all users of the system.
4. A directory containing accounting information, named ACCOUNTS.
5. A directory of programs, named PROGRAMS.

Suppose three coworkers, Joe, Sue, and Mary, each have their own directories, which are subdirectories of the USER directory. Sue has a subdirectory named FORMS. Sue and Mary have files in their directories, each named SALES.MAY. Mary's SALES.MAY file is unrelated to Sue's.

This organization of files and directories is not important if you only work with files in your own directory, but if you work with someone else, or on several projects at once, the multilevel directory structure becomes handy. For example, you could get a list of the files in Sue's FORMS directory by typing:

```
DIR \USER\SUE\FORMS <Enter>
```

Note that a backslash mark (\) separates directories from other directories and files. The first slash indicates the separation from the root directory.

To find out what files Mary has in her directory, you could type:

```
DIR \USER\MARY <Enter>
```

This command tells MS-DOS to travel down the directory structure from the root to the USER directory, and then to the MARY directory, and to display all file names in the MARY directory.

## 1.7 Paths

When you use multilevel directories, you must tell MS-DOS where the files are located in the directory structure. Both Mary and Sue, for example, have files named SALES.MAY. Each will have to tell MS-DOS in which directory her file resides when she wants to use it. They can do this by giving MS-DOS a path to the file.

### 1.7.1 Path Format

A path is a sequence of directory names. Each directory is separated from the previous directory by a backslash (\).

The general format of a path is:

[directoryname]\[directoryname...]

A path can contain any number of directory names. If a path name begins with a slash, MS-DOS searches for the file beginning at the root (or top) of the directory structure. Otherwise, MS-DOS begins at the working (current) directory and searches downward from there.

#### NOTE

Some MS-DOS commands, such as DIR and PRINT, cannot accept a path, filename, and extension of more than 64 characters, even though you can create directories longer than this limit. You must change directories to access these files.

The path of Sue's SALES.MAY file is:

`\USER\SUE`

To access the file, Sue must add the filename and extension to the path such as:

`\USER\SUE\SALES.MAY`

The path of Mary's SALES.MAY file is:

`\USER\MARY`

To access the file, Mary must also add the filename and extension to the path such as:

`\USER\MARY\SALES.MAY`

When you are in your working directory, a file name and its corresponding path may be used interchangeably. Some sample names are:

<code>\</code>	Indicates the root directory.
<code>\PROGRAMS</code>	Directory under the root directory containing program files.
<code>\USER\MARY\FORMS\1A</code>	A typical full path. This one is file named 1A in the directory named FORMS belonging to Mary.
<code>SALES.MAY</code>	Name of a file in the working directory.

Each directory that has subdirectories beneath it is called a *parent directory*. MS-DOS provides special shorthand notations for the working directory and the parent directory (one level up) of the working directory:

- A single period is the name of the working directory in all multilevel directory listings.  
  
MS-DOS automatically creates this entry when a directory is made.
- • Two periods are the name of the working directory's parent directory (one level up).

If you type:

```
DIR .. <Enter>
```

MS-DOS lists the files in the parent directory of your working directory.

If you type:

```
DIR ..\..\ <Enter>
```

MS-DOS lists the files in the parent's *parent directory*.

### 1.7.2 Paths and External Commands

External commands reside on disks as program files. They must be read from the disk before they can execute. (External commands are detailed in Section 2.2.)

When you are working with more than one directory, it is convenient to put all MS-DOS external commands into a separate directory so they do not clutter your other directories.

When you issue an external command, MS-DOS immediately checks your working directory to find that command. You must tell MS-DOS in which directory these external commands reside. This is done with the PATH command.

For example, if you are in a working directory named \BIN\PROG, and all MS-DOS external commands are in \BIN, you must tell MS-DOS to choose the \BIN path to find the FORMAT command. The command:

```
PATH \BIN
```

tells MS-DOS to search in your working directory and the \BIN directory for all commands. You only have to specify this path once to MS-DOS during each computer session. If you want to know what the working path is, type:

```
PATH <Enter>
```

The working path will display.

You can establish a search path by including the PATH command in a special file named AUTOEXEC.BAT. Refer to Section 2.7 for more information on the AUTOEXEC.BAT file.

For more information on the PATH command, refer to Section 3.3.



### 1.7.3 Paths and Internal Commands

Internal commands are the simplest, most commonly used commands. They execute immediately because they are loaded into your computer's memory when you start MS-DOS. (For more information on internal commands, refer to Section 2.1.)

Some internal commands can use paths. The commands, COPY, DIR, and DEL have greater flexibility when you specify a path after the command.

For example:

```
COPY path1 path2
```

All files in the path1 directory are copied to the path2 directory.

For example:

```
DEL path1 <Enter>
```

Since path1 is a directory, all the files in path1 are deleted. The prompt "Are you sure (Y/N)?" is displayed if you try to delete a path. Type Y (for Yes) to complete the command, or type N (for No) to stop the command.

To display the directory of path, type:

```
DIR path <Enter>
```

## 1.8 Using Directories

The following sections describe how to display, change, and delete your working directory. You will also learn how to create directories and subdirectories.

### 1.8.1 How to Display Your Working Directory

All commands are executed while you are in your working directory. You can find out the name of your working directory by issuing the CHDIR (Change Directory) command without a path. For example, if your working directory is \USER\JOE, when you type:

```
CHDIR <Enter>
```

you would see:

```
A:\USER\JOE
```

This is your working drive (A:) and working directory (\USER\JOE).

#### NOTE

You can use the letters CD for the CHDIR command to save time. The command CD USER\JOE is the same as CHDIR \USER\JOE.

If you now want to see what is in the \USER\JOE directory, you can issue the DIR command. The following is an example of the display you might receive from the DIR command for a subdirectory:

Volume in drive A has no ID  
Directory of A:\USER\JOE

.	<Dir>		12-02-85	10:09a
..	<Dir>		12-02-85	10:09a
TEXT	<Dir>		12-02-85	10:09a
FILE1	COM	5243	12-02-85	11:30a

4 File(s) 836320 bytes free

A volume ID for this disk was not assigned when the disk was formatted. (For information on assigning a volume ID to a disk, turn to the FORMAT command in Section 3.3)

MS-DOS lists both files and directories in this output. As you can see, Joe has a subdirectory named TEXT. The "." means the working directory (\USER\JOE) and the ".." is short for the parent directory (\USER). FILE1.COM is a file in the \USER\JOE directory. All of these directories and files are on the disk in drive A.

#### NOTE

MS-DOS considers a directory to be just a special type of file. Therefore, you cannot give a subdirectory the same name as a file in that directory. For example, if you have a path \BIN\USER\JOE where JOE is a subdirectory, you cannot create a file named JOE in the \BIN\USER directory.

### 1.8.2 How to Create a Directory

To create a subdirectory in your working directory, issue the MKDIR (Make Directory) command. For example, to create a new directory named NEWDIR under your working directory, type:

```
MKDIR newdir <Enter>
```

After MS-DOS runs this command, a new directory exists under your working directory. You can also create directories anywhere in the directory structure by specifying MKDIR and then the path of the directory you want to create. However, the path cannot be over 64 characters long.

MS-DOS automatically creates the "." and ".." entries in the new directory.

To put files in the new directory, use the MS-DOS line editor, EDLIN. You can also create and save files if you have purchased a word processing program such as Microsoft<sup>®</sup> Word.

### 1.8.3 How to Change Your Working Directory

To change from your working directory to another directory type CHDIR and a path. For example, type:

```
CHDIR \USER <Enter>
```

MS-DOS changes the working directory to \USER. You can specify any path after CHDIR to *travel* around the directory structure. The command:

```
CHDIR .. <Enter>
```

always puts you in the parent directory of your working directory.

### 1.8.4 How to Delete a Directory

To delete a directory in the structure, issue the RMDIR (Remove Directory) command. For example, to remove the NEWDIR directory from the working directory, type:

```
RMDIR NEWDIR <Enter>
```

The NEWDIR directory must be empty except for the "." and ".." entries before it can be removed; this prevents you from accidentally deleting files and directories. You can delete any directory by specifying its path name.

To remove the \USER\JOE directory, make sure it has only the "." and ".." entries, then type:

```
RMDIR \USER\JOE <Enter>
```

To remove all the files in a directory (except the "." and ".." entries), type DEL and then the path name of the directory. For example, to delete all files in the \USER\SUE directory, type:

```
DEL \USER\SUE <Enter>
```

MS-DOS prompts:

```
Are you sure (Y/N)?
```

If you are sure you want to delete all the files in the directory, type Y (for Yes). Type N (for No) to stop the command.

You cannot delete the "." and ".." entries. They are created by MS-DOS as part of the multilevel directory structure.

## 1.9 How To Rename A Directory

There is no command to rename a directory in MS-DOS. You can, however, rename a directory that has no subdirectories by following these steps:

1. Create a new directory with the new name using the MKDIR command.
2. Copy all files from the old directory to the new directory with the COPY \*.\* command.
3. Delete the contents of the old directory with the DEL \*.\* command.
4. Delete the old directory with the RMDIR command.

Example:

To rename the \USER\JOE directory to \USER\SUE, type:

```
MKDIR \USER\SUE <Enter>
```

```
COPY \USER\JOE\*.* \USER\SUE <Enter>
```

```
DEL \USER\JOE\*.* <Enter>
```

(Type Y in response to the prompt "Are you sure?")

```
RMDIR \USER\JOE <Enter>
```

## 1.10 Summary Of Commands

Command	Purpose	Example
DIR	Displays a directory	DIR A:
CHKDSK	Checks disks	CHKDSK B:
PATH	Sets MS-DOS search path	PATH \DEB\GAMES
CHDIR	Displays working directory; changes directories	CHDIR \USER \BILL
MKDIR	Makes a new directory	MKDIR \USER \SALLY
RMDIR	Removes a directory	RMDIR \BIN\GAMES

Learning About Commands





This section covers the following topics:

- What is a command?
- Types of MS-DOS commands.
- Command options.
- Things you should know about commands.
- Input and output.
- Batch processing.
- The AUTOEXEC.BAT file.
- How to create a batch file with replaceable parameters.

## 2.1 What Is a Command?

A command is a way of communicating with the computer. By typing MS-DOS commands at your terminal, you can ask the computer to perform useful tasks. There are MS-DOS commands that:

- Compare, copy, display, delete, and rename files.
- Copy and format disks.
- Execute programs.
- Analyze and list directories.
- Enter date, time, and remarks.
- Set various printer and screen options.
- Copy MS-DOS system files to another disk.
- Request MS-DOS to wait for a specific periods of time.

## 2.2 Types Of MS-DOS Commands

There are two types of MS-DOS commands: internal commands, and external commands.

### 2.2.1 Internal Commands

Internal commands are the simplest, most commonly used commands. You cannot see these commands when you do a directory listing on your MS-DOS disk; they are part of a large file named COMMAND.COM. When you type Internal commands, they execute immediately. The following Internal commands are described in Section 3.3:

BREAK	DEL	MKDIR	SET
CHDIR	DIR	PATH	SHIFT
CLS	ECHO	PAUSE	TIME
COPY	EXIT	PROMPT	TYPE
CTTY	FOR	REM	VER
DATE	GOTO	REN	VERIFY
IF	RMDIR	VOL	

## 2.2.2 External Commands

External commands are on disks as program files. They must be read from a disk before they can execute.

Any file name with a file name extension of .COM, .EXE or .BAT is considered an external command. For example, programs such as FORMAT.COM and DISKCOPY.COM are external commands.

Because all external commands are files, you can create commands and add them to MS-DOS. Programs you create with most languages (including assembly language) are .EXE (executable) files.

When you type an external command, do not include its file name extension. The following external commands are described in Section 3.3:

ASSIGN	FDISK	PRINT
ATTRIB	FIND	RECOVER
BACKUP	FORMAT	REPLACE
CHKDSK	GRAFTABL	RESTORE
COMMAND	GRAPHICS	SHARE
DISKCOMP	JOIN	SORT
DISKCOPY	LABEL	SUBST
EXE2BIN	MODE	SYS
	MORE	TREE

## 2.3 Command Options

You can specify options in your commands to give MS-DOS extra information. If you do not include some options, MS-DOS provides a value called a *default value*. Refer to individual command descriptions in Section 3.3 for the default values.

Square brackets ([ ]) mean that an item is optional. Uppercase letters mean you type in the text exactly as shown. Lowercase letters mean that you substitute a value for the item shown.

Note that the drive name is not required unless you need to indicate to MS-DOS which disk to search for a specific file.

The following is the format of all MS-DOS commands:

Command [options...]

where *options...* can be one of the following:

drive	The disk drive name.
filename	Any name for a disk file, including the file name extension if there is one. The <i>filename</i> option does not refer to a device or to a disk drive name.
path	A path in the following general format:  [directory]\[directory..]
switches	Options that control MS-DOS commands. They are preceded by a forward slash (for example, /P).
arguments	Provide more information to MS-DOS commands. You usually choose between arguments, for example, ON or OFF.

## 2.4 Things You Should Know About Commands

The following information applies to all MS-DOS commands:

1. Commands are usually followed by one or more options.
2. Commands and options can be typed in uppercase, lowercase letters, or any combination of the two.
3. Commands and options must be separated by certain characters or spaces. Because they are easiest, use the space or comma to separate commands from options. For example:

```
DEL NEWFILE.OLD NEWFILE.TXT
```

```
RENAME,THISFILE THATFILE
```

You can also use the semicolon (;), the equal sign (=), or the <Tab> key between MS-DOS commands and their options.

This manual shows a space between commands and options, such as:

```
COPY MEM01 MEM02
```



4. Commands take effect only after you have pressed `< Enter >`.
5. When instructions say "Press any key," you can press any alpha (A-Z) or numeric (0-9) key, or `< Space >`.
6. You must include the file name extension when referring to a file that has one.
7. You can stop commands while they are running by pressing `< Ctrl > - < C >`.
8. When commands produce a large amount of output on the screen, the display automatically scrolls to the next screen. You can press `< Ctrl > - < S >` to suspend the scrolling. Press `< Ctrl > - < S >` again to view the rest of the display.
9. MS-DOS editing and function keys can be used when typing commands. Refer to Section 4 for a complete description of these keys.
10. Disk drives are referred to as source drives and destination drives. A source drive is the drive from which you transfer information. A destination drive is the drive to which you transfer the information.

## 2.5 Input and Output

MS-DOS always assumes input comes from the keyboard and output goes to the screen. However, you can redirect the flow of command input and output. Input can come from a file rather than a keyboard, and output can go to a file or to a line printer instead of to the screen.

Also, you can create *pipes* to allow output from one command to become the input to another. Redirection and pipes are discussed in the next sections.

### 2.5.1 How to Redirect Your Output

Most commands send output to your terminal screen. You can send output information to a file by using a greater-than sign (>) in your command.

For example, the command:

```
DIR <Enter>
```

displays a directory listing of the disk in the default drive on the screen. The same command can send this output to a file named NEWFILES by typing the output file on the command line following a greater-than sign (>):

```
DIR >NEWFILES <Enter>
```

If the file NEWFILES doesn't exist, MS-DOS creates it and stores your directory listing in it. If NEWFILES exists, MS-DOS overwrites what is in the file with the new data.

If you want to append your directory or a file to another file (instead of replacing the entire file), you can specify two greater-than signs (> >). For example, the command:

```
DIR >>NEWFILES
```

appends your directory listing to a file named NEWFILES. If NEWFILES doesn't exist, MS-DOS creates it.

It is often useful to have input for a command come from a file rather than from a terminal. This is possible in MS-DOS by specifying a less-than sign (<) in your command.

For example, the command:

```
SORT <NAMES >LIST1
```

sorts the file NAMES and sends the sorted output to a file named LIST1. (For more information on the SORT command, see Section 3.3.)

### 2.5.2 Filters

A filter is a command that reads your input, transforms it, and then produces output. The data is *filtered* by the program.

Since filters can be put together in many different ways, a few filters can take the place of many specific commands.

MS-DOS filters include FIND, MORE, and SORT. Their functions are described below:

FIND	Searches for some text in a file.
MORE	Displays text one screen at a time.
SORT	Sorts text.

You can see how these filters are used in the next section.

### 2.5.3 Command Piping

If you want to give more than one command at a time to the system, you can *pipe* commands to MS-DOS. For example, sometimes you might want the output of one program sent as the input to another program. A typical case would be a program that produces output in columns. You might want to have these columns sorted.

Piping is done by separating commands with the pipe symbol, the vertical bar (|). For example, the command:

```
DIR | SORT
```

displays an alphabetically-sorted listing of your directory on the screen. The vertical bar causes all output generated by a command on the left side of the bar to be sent to the right side of the bar for processing.

You can also use piping when you want to send output to a file. If you want your directory sorted and sent to a new file (for example, DIREC.FIL), you could type:

```
DIR | SORT >DIREC.FIL
```

MS-DOS creates a file named DIREC.FIL on your default drive. DIREC.FIL contains a sorted listing of the directory on the default drive, since no other drive was specified in the command. To specify a drive other than the default drive (for example, drive B), type:

```
DIR | SORT >B:DIREC.FIL
```

This sends the sorted data to a file named DIREC.FIL on drive B.

A *pipeline* can consist of more than two commands. For example:

```
DIR | SORT | MORE
```

sorts your directory, shows it to you one screen at a time, and puts --MORE-- at the bottom of your screen when there is more output to be seen.

## 2.6 Batch Processing

Often you may find yourself typing the same sequence of commands over and over to perform some common task. With MS-DOS, you can put the command sequence into a special file called a *batch file*, and execute all the commands simply by typing the name of the batch file.

*Batches* of your commands in such files are processed as if they were typed from the keyboard. Each batch file must be named with the .BAT extension, and is executed by typing the file name without its extension.

You can create a batch file by using EDLIN, the line editor, or by typing the Copy command. Refer to Section 2.7.1 for more information on using the Copy command to create a batch file.

Two MS-DOS commands are used only in batch files: REM and PAUSE. REM allows you to include comments in your batch files without these lines being executed as commands. PAUSE prompts you with an optional message and permits you to either continue or stop the batch process at a given point. REM and PAUSE are described in detail in Section 3.3.

Batch processing is useful if you want to execute several MS-DOS commands with one batch command. For example, a batch file to format and check a new disk might look like this:

```
REM   THIS IS A FILE TO CHECK NEW DISKS
REM   IT IS NAMED NEWDISK.BAT
PAUSE  INSERT NEW DISK IN DRIVE B:
FORMAT B:
CHKDSK B:
```

To execute this .BAT file after you have entered and saved it as NEWDISK.BAT, simply type the file name without the .BAT extension:

```
NEWDISK <Enter>
```

The result is the same as if the lines in the .BAT file were entered from the keyboard as individual commands.

The following list contains information you should read before you run a batch process with MS-DOS:

1. Only the file name should be entered to execute the batch file. Do not type the file name extension.
2. If you press < Ctrl > - < C > when the batch file is running, this prompt appears:

Terminate batch job (Y/N)?

If you press Y, the remaining commands in the batch file are ignored and the MS-DOS prompt (usually A > ) appears.

If you press N, the current command aborts and batch processing continues with the next command in the file.

3. If you remove the disk that contains a batch file being run, MS-DOS will prompt you to insert it again before the next command can be read.
4. The last command in a batch file can be the name of another batch file. This allows you to call one batch file from another when the first is finished.
5. You can redirect output in a batch file with the > and < symbols. However, you cannot use piping (the symbol |) in a batch file. Refer to Section 2.5.1 and Section 2.5.3 for more information.



## 2.7 The AUTOEXEC.BAT File

An AUTOEXEC.BAT file allows you to automatically execute programs when you start MS-DOS. This is useful when you want to run a specific package under MS-DOS, and when you want MS-DOS to execute a batch program each time you start your computer.

You can avoid loading two separate disks to perform either of these tasks by using an AUTOEXEC.BAT file.

When you start your computer, MS-DOS searches the disk for a file named AUTOEXEC.BAT. The AUTOEXEC.BAT file is a batch file that is automatically executed each time you start the system. It must be located in the root directory.

If MS-DOS finds the AUTOEXEC.BAT file, the file is immediately executed and the date and time prompts are bypassed.

If MS-DOS does not find an AUTOEXEC.BAT file when you first load the MS-DOS disk, then the date and time prompts appear.

Figure 2-1 shows how MS-DOS uses the AUTOEXEC.BAT file.

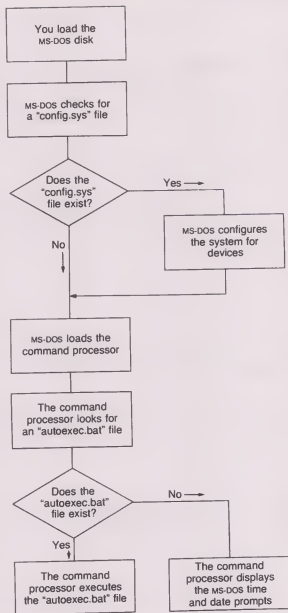


Figure 2-1. How MS-DOS Uses the AUTOEXEC.BAT File.

If, for example, you want to automatically load GW-BASIC® and run a program called MENU each time you start MS-DOS, you could create an AUTOEXEC.BAT file as follows:

1. Type:

```
COPY CON AUTOEXEC.BAT <Enter>
```

This statement tells MS-DOS to copy the information from the console (keyboard) into the AUTOEXEC.BAT file.

Note that the AUTOEXEC.BAT file must be created in the root directory of your MS-DOS disk.

2. Now type:

```
GW BASIC MENU
```

This statement goes into the AUTOEXEC.BAT file. It tells MS-DOS to load BASIC and run the MENU program whenever MS-DOS is started.

3. Press <Ctrl>-<Z> then press <Enter> to put the command GW BASIC MENU in the AUTOEXEC.BAT file.
4. The MENU program will now run automatically whenever you start MS-DOS.

To run your own GW-BASIC program, type the name of your program in place of MENU in the second line of the example. You can enter any MS-DOS command or series of commands in the AUTOEXEC.BAT file.

## 2.8 How To Create A Batch File With Replaceable Parameters

Sometimes you want to create a program and run it with different sets of data. These data can be stored in various MS-DOS files.

When used in MS-DOS commands, a parameter is an option you define. With MS-DOS, you can create a batch (.BAT) file with dummy (replaceable) parameters. These parameters, named %0-%9, can be replaced by values supplied when the batch file executes.

For example, when you type the command line:

```
COPY CON NEWFILE.BAT
```

the next lines you type are copied from the console to a file named NEWFILE.BAT on the default drive:

```
DEL %3.DOC  
COPY %1.DOC + %2.DOC %3.DOC  
PRINT %3.DOC
```

Now, press <Ctrl>-<Z> and then <Enter>. MS-DOS responds with this message:

```
1 File(s) copied
```

The file NEWFILE.BAT, which consists of three commands, now resides on the disk in the default drive.

The dummy parameters %1 and %2 are replaced sequentially by the parameters you supply when you execute the file.

For example, if you now type:

**NEWFILE FILE1 FILE2 FILE3 <Enter>**

MS-DOS substitutes FILE1 for %1, FILE2 for %2, and FILE3 for %3. Thus, it first erases FILE3.DOC, creates a new FILE3.DOC by appending FILE2.DOC to FILE1.DOC, then prints the resulting FILE3.DOC.

If you include the dummy parameter %0, it is always replaced by the drive name, if specified, and the file name of the batch file (for example, NEWFILE).

### NOTES

1. Up to 10 dummy parameters (%0-%9) can be specified.

Refer to the MS-DOS SHIFT command in Chapter 3 if you want to specify more than 10 parameters.

2. If you use the percent sign as part of a file name within a batch file, you must type it twice. For example, to specify the file ABC%.EXE, you must type it as ABC%%.EXE in the batch file.

To execute the batch file NEWFILE.BAT and to specify the parameters to replace the dummy parameters, you must enter the batch file name (without its extension) followed by the parameters you want MS-DOS to substitute for %1, %2, and %3.

Remember that the file NEWFILE.BAT consists of 3 lines:

```
DEL %3.DOC
COPY %1.DOC + %2.DOC %3.DOC
PRINT %3.DOC
```

To execute the NEWFILE batch process, type:

```
NEWFILE A:MEMO1 B:MEMO2 B:BIGFILE <Enter>
```

A:MEMO1 is substituted for %1, B:MEMO2, for %2, and B:BIGFILE for %3.

The result is the same as if you had typed each of the commands in NEWFILE with their parameters, as follows:

```
DEL B:BIGFILE
COPY A:MEMO1 + B:MEMO2 B:BIGFILE
PRINT B:BIGFILE
```

The following shows how MS-DOS replaces each of the above parameters:

BATCH	PARAMETER 1	PARAMETER 2	PARAMETER 3
FILENAME	(%1)	(%2)	(%3)
newfile	memo1.doc	b:memo2.doc	b:bigfile

Remember that if you specify the dummy parameter %0 in your batch file, it is always replaced by the drive name (if specified) and the file name of the batch file. If you do not need to refer to the batch file, start your dummy parameters at number %1.

## 2.9 Command Summary

Command	Purpose	Example
REM	Adds comment line. This is a test file for batch files	REM
PAUSE	Suspends execution Insert blank disk of a batch file	PAUSE
FIND	Searches for string of text in a specified file	FIND WHO NEWFILE.TXT
MORE	Pages through a type file 23 lines at a time	MEMO.TXT MORE
SORT	Sorts text	SORT NAMES.FIL

MS-DOS Commands





This section explains how to use each MS-DOS command.

### 3.1 About Commands

You should keep the following points in mind as you format MS-DOS commands:

1. You can type commands in any combination of uppercase or lowercase letters; MS-DOS changes them to uppercase letters.
2. Supply the text for any items shown in lowercase. For example, you should type the name of your file when *filename* is shown in the format.
3. Items in square brackets ( [ ] ) are optional. If you want to include optional information, do not include the square brackets, only the information within the brackets.
4. An ellipsis (...) means you can repeat an item as many times as necessary.
5. For some commands, the syntax is too long to fit onto one line, so it is split into two (or more) lines. You, however, must give the command on one line.

6. You must include all punctuation where shown (except for square brackets), such as commas, equal signs, question marks, colons, or slashes.
7. Unlike previous versions of MS-DOS, you can specify a path to run a specific command or program. For example, if you move the ASSIGN program to a directory named \BIN, you can type:

\BIN\ASSIGN C=E

8. Many commands manipulate *strings* of text. A *string* is a group of characters; it can include letters, numbers, spaces, and all other characters.

Searching for a particular word in a file is a common use of a string.

9. The format instructions for each command are based on the following:

*d:* Is the name of the disk drive.

*path* Is the name of a directory.

*filename* is the title of the file.

*.ext* is the filename extension.

## 3.2 About This Chapter

The commands in this chapter are divided into two sections:

- Common MS-DOS commands.
- Batch commands.

Commands are either external or internal. Each external command resides on disk as a file. External commands have file name extensions of .BAT, .COM, or .EXE. Internal commands are included in a file named COMMAND.COM on the MS-DOS disk.

Some MS-DOS commands do not work over a computer network. If you try to use these commands, MS-DOS displays this error message:

Cannot <command> to a network device

where *command* is the name of the command you typed.

The commands that do not work over a network (on a shared or remote device) are:

CHKDSK  
DISKCOMP  
DISKCOPY  
FORMAT  
LABEL  
RECOVER  
SUBST  
SYS

### 3.3 MS-DOS Commands

The following MS-DOS commands are described in this chapter. Synonyms for commands are in parentheses.

ASSIGN	Assigns a drive letter to a different drive.
ATTRIB	Sets or displays attributes of a file.
BACKUP	Backs up files from a hard disk.
BREAK	Sets <Ctrl> - <C> check.
CHDIR	Changes directories; prints working directory (CD).
CHKDSK	Scans the directory of the default or designated drive and checks for consistency.
CLS	Clears the screen.
COMMAND	Processes internal MS-DOS commands.
COMP	Compares the contents of two files.
COPY	Copies the file(s) specified.
CTTY	Changes the console TTY.
DATE	Displays and sets the date.
DEL	Deletes the file(s) specified (Erase).

DIR	Lists requested directory entries.
DISKCOMP	Compares disks.
DISKCOPY	Makes a copy of a disk.
EXE2BIN	Converts executable files to binary format.
EXIT	Exits command processor and returns to previous level.
FDISK	Partitions hard disk.
FIND	Searches for a constant string of text.
FORMAT	Formats a disk to receive MS-DOS files.
GRAFTABL	Loads a table of graphics characters.
GRAPHICS	Prepares MS-DOS for printing graphics.
JOIN	Joins a disk drive to a path name.
KEYBxx	Replaces resident keyboard program with non-U.S. keyboard programs.
LABEL	Labels disks.
MKDIR	Makes a directory (MD).
MODE	Modifies screen, communications port, and printer port parameters.

MORE	Displays output one screen at a time.
PATH	Sets a command search path.
PRINT	Prints files.
PROMPT	Assigns a default prompt.
RECOVER	Recovers a bad disk or file.
REN	Renames first file as second file (Rename).
REPLACE	Add or replace files on a target diskette with files from a source diskette.
RESTORE	Restores backed up files.
RMDIR	Removes a directory (RD).
SELECT	Sets date and time format.
SET	Sets one string value to another in the environment, or displays the environment.
SHARE	Installs file sharing and locking.
SORT	Sorts data forward or backward.
SUBST	Substitutes a string for a path name.
SYS	Transfers MS-DOS system files from one drive to the drive specified.

TIME	Displays and sets the time.
TREE	Displays directory and file names.
TYPE	Displays the contents of a file.
VER	Prints MS-DOS version number.
VERIFY	Verifies all writes to a disk.
VOL	Displays the volume ID.
XCOPY	Copies files and subdirectories.

#### **Batch Commands**

ECHO	Turns batch file echo feature on/off or displays setting.
FOR	Batch command extension.
GOTO	Batch command extension.
IF	Batch command extension.
PAUSE	Pauses for input in a batch file.
REM	Displays a comment in a batch file.
SHIFT	Increases number of replaceable parameters in batch process.

Each command is described in detail in the following pages.



## ASSIGN

---

**Purpose:** Assigns a drive letter to a different drive.

**Type:** External

**Format:** ASSIGN[[X]=[Y]]

where:

x is the drive to which reads and writes are currently sent.

y is the drive to which you want reads and writes sent.

**Remarks:**

The equal sign ( = ) is optional.

You do not have to type the colon after the drive letters x and y.

To reset all drives back to their original assignments, type:

ASSIGN

The ASSIGN command allows you to perform disk operations on drives other than A and B for programs that only use those two drives. The command

ASSIGN A=C B=C

enables you to use drives other than A and B, such as a hard disk (C). All references to drives A and B now go to drive C.

#### NOTE

Do not use the ASSIGN command with the BACKUP or PRINT commands or during normal use of MS-DOS. The ASSIGN command hides the true device type from commands that require actual drive information. The FORMAT and DISKCOPY programs ignore any drive reassignments.

#### Messages:

Incorrect parameter

One of the options you specified is wrong.

## ATTRIB

---

**Purpose:** Sets or resets the read-only and archive attributes of a file; displays the attributes of a file.

**Type:** External

**Format:** ATTRIB[+R|-R] [+A|-A]  
[D:][path] filename[.ext]

where:

+R sets the read-only attribute of a file.

-R disables read-only mode.

+A sets the archive attribute of a file.

-A clears the archive attribute of a file.

*d:path filename.ext* specifies the file you want to reference.

**Remarks:**

ATTRIB will not display and cannot modify the attributes of hidden or system files.

If an application opens a file with read and write permission, ATTRIB forces read-only mode to allow file sharing over a network.

The BACKUP, RESTORE, and XCOPY commands use the archive attribute to perform a selective BACKUP, RESTORE, or XCOPY on files that have been modified. You can use the +A and -A options to select files you want to back up with the BACKUP /M switch or copy with the XCOPY /M switch.

To display the attribute of a specific file, type:

```
ATTRIB PATH FILENAME.EXT <Enter>
```

You can specify the wildcard characters \*.\* to print the attributes of all files on a specific drive.

**Example:**

The following example makes the file named MYFILE.TXT read-only:

```
ATTRIB +R MYFILE.TXT
```

## BACKUP

---

- Purpose:** Copies one or more files from one disk to another.
- Type:** External
- Caution:** BACKUP destroys all existing files on the archive disk unless you use the /A (append) switch.
- Format:** BACKUP d1:[path][filename[.ext]]  
d2:[/S][/M][/A]/[/D:mm-dd-yy]

where:

*d1: path filename.ext* is the file you want to back up. *d1:* is referred to as the source drive.

*d2:* is the archive disk. *d2:* is referred to as the target drive.

### Switches:

- /S** backs up files in all subdirectories as well as files in the specified directory.
- /M** copies only files that have changed since the last backup.
- /A** adds files to be copied to the files on the disk. /A prevents BACKUP from erasing existing files on the backup disk.
- /D** copies files created or modified on or after the date entered.

**Remarks:**

The source drive (d1:) must be different than the target drive (d2:).

If you do not enter a filename, BACKUP copies all files in your current directory.

If you share files, you can only back up files you can access.

BACKUP files can only be used in RESTORE operations. The RESTORE command is explained later in this section.

Label each BACKUP diskette and record the date and a number, so you can RESTORE in sequential order.

Do not use BACKUP with drives you have reassigned with ASSIGN or SUBST commands. Drive substitutions can hide the real drive letter BACKUP uses.

Do not use BACKUP while a JOIN command is in effect, because the directory structure may be invalid when you restore.

The BACKUP exit codes are:

- 0 Normal completion.
- 1 No files to copy.
- 2 Unable to copy some files due to sharing conflicts.
- 3 Aborted, user pressed <Ctrl>-<C>.
- 4 Error caused BACKUP to terminate.

Use the exit codes with the batch processing subcommand IF to control subsequent error level processing.

### Examples:

To copy all files on hard disk drive C:, type:

```
BACKUP C:\ A:/S
```

To copy all .DAT files from the current directory to the diskette in drive A:, type:

```
BACKUP C:* .DAT A:
```

To copy all modified files in the current directory, type:

```
BACKUP C:*.* A:/M
```

### Messages:

Not a BACKUP disk

If you use the /A switch, BACKUP only copies to a disk with existing BACKUP files.

## BREAK

---

**Purpose:** Determines which activities are stopped when either <Ctrl>-<C> or <Ctrl>-<Break> is pressed.

**Type:** Internal

**Format:** BREAK [ON]  
  
or  
  
BREAK [OFF]

**Remarks:**

Depending on the program you are running, pressing <Ctrl>-<C> or <Ctrl>-<Break> can stop an activity (for example, to stop sorting a file). Normally, MS-DOS checks to see if <Ctrl>-<C> or <Ctrl>-<Break> has been pressed while reading from the keyboard, writing to the screen, or to a printer.

Setting BREAK to ON allows <Ctrl>-<C> and <Ctrl>-<Break> checking to be extended to other functions such as disk reads and writes.

To check for <Ctrl>-<C> or <Ctrl>-<Break> during screen, keyboard, and printer reads and writes only, set BREAK to OFF.

To find out how BREAK is currently set, type:

BREAK<Enter>



## CHDIR

---

**Purpose:** Changes directory to a different path; displays working directory.

**Type:** Internal

**Format:** CHDIR [path]

or

CD [path]

where *path* specifies the new working directory.

**Remarks:**

If your working directory is \BIN\USER\JOE and you want to change your path to another directory (such as \BIN\USER\JOE\FORMS), type:

```
CHDIR \BIN\USER\JOE\FORMS<Enter>
```

MS-DOS will put you in the new directory.

The following command puts you in the parent directory of your working directory.

```
CHDIR ..
```

CHDIR specified without a path displays your working directory. If your working directory is \BIN\USER\JOE on drive B, and you type:

```
CHDIR<Enter>
```

MS-DOS displays:

```
B:\BIN\USER\JOE
```

issuing the CHDIR command in this way is useful when you need to know the name of your working directory.

The command:

```
CHDIR B:
```

displays the working directory on drive B.

The command:

```
CD \<Enter>
```

changes the working directory to the root directory.

# CHKDSK

---

**Purpose:** Scans the disk in the specified drive and checks it for errors.

**Type:** External

**Format:** CHKDSK [d:] [path][filename[.ext]]  
[/F] [/V]

where *d: path filename.ext* specifies the file name.

**Switches:**

- /F fixes any errors found in the directory.
- /V displays messages while CHKDSK is running.

**Remarks:**

Run CHKDSK occasionally on each disk to check for errors. If any errors are found, CHKDSK displays the appropriate error messages, if any, followed by a status report.

A sample status report looks like this:

```
Volume DOSDISK      created Jan 12, 1986 3:21p

          362496      bytes total disk space
          38912      bytes in 2 hidden files
           1024      bytes in 2 directories
         282624      bytes in 42 user files
          39936      bytes available on disk

          524288      bytes total memory
          487248      bytes free
```

If you type a file name after CHKDSK, MS-DOS displays a status report for the disk and for the individual file.

You can redirect the output from CHKDSK to a file if you want to save the status report for future use. Simply type:

```
CHKDSK A:>FILENAME
```

The errors are sent to the specified file name. Do not use the /F switch if you redirect CHKDSK output.

## Messages:

CHKDSK may display the following messages:

(.)(..) does not exist

This is an informational message from CHKDSK. This message indicates that either the "." or the ".." directory entry is invalid.

(filename) contains non-contiguous blocks

The file or files you named are not written contiguously on the disk.

All specified file(s) are contiguous

The file or files you named are all written sequentially on disk.

The following errors are corrected automatically if you specify the /F switch:

Allocation error, size adjusted

An invalid cluster number was found in the FAT. The file was truncated at the end of the last valid cluster.

Entry has a bad attribute (or size or link)

This message may be preceded by one or two periods to indicate which subdirectory is invalid. If you have specified the /F switch, CHKDSK attempts to correct the error.

Errors found, F parameter not specified

Corrections will not be written to disk. You must specify the /F switch if you want the errors corrected by CHKDSK.

First cluster number is Invalid

Entry truncated

The file directory entry contains an Invalid pointer to the data area. If specified the /F switch, the file is truncated to a zero-length file.

X lost clusters found in y chains

Convert lost chains to files (Y/N)?

If you respond Y to this prompt, and if you specified the /F switch, CHKDSK creates a file in the root directory for you to resolve this problem (files created by CHKDSK are named FILEnnnn.CHK). If you did not specify the /F switch, CHKDSK does nothing.

If you respond N to this prompt and have specified the /F switch, CHKDSK displays:

X bytes disk space freed

If you respond N to this prompt and have not specified the /F switch, CHKDSK displays:

X bytes disk space would be freed

MS-DOS cannot correct the following errors returned by CHKDSK, even if you specified the /F switch. You must take action to correct the situation:

(filename) is cross-linked on cluster

Make a copy of the file you want to keep, and then delete both files that are cross-linked on the same cluster.

Cannot CHDIR to (filename)

Tree past this point not processed. CHKDSK is traveling the tree structure of the directory and is unable to proceed to the specified directory. All subdirectories and files under this directory are not verified.

Cannot CHDIR to root

Processing cannot continue

CHKDSK is traveling the tree structure of the directory and is unable to return to the root directory. CHKDSK is not able to continue checking the remaining subdirectories to the root. Try to restart MS-DOS. If this error persists, the disk is unusable.

Cannot recover . entry, processing continued

The "." entry (working directory) is defective.

Cannot recover .. entry

The ".." (parent directory) entry is defective.

Directory is totally empty, no . or ..

The specified directory does not contain references to working and parent directories. Delete the specified directory and recreate it.

**Disk error reading FAT**

One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

**Disk error writing FAT**

One of your File Allocation Tables has a defective sector in it. MS-DOS will automatically use the other FAT. It is a good idea to copy all your files onto another disk.

**Incorrect DOS version**

You cannot run CHKDSK on versions of MS-DOS before 3.0.

**Insufficient memory****Processing cannot continue**

There is not enough memory in your machine for CHKDSK to process this disk. You must obtain more memory to run CHKDSK.

**Insufficient room in root directory**

Erase files in root and repeat CHKDSK. CHKDSK cannot run until you delete files in the root directory.

**Invalid drive specification**

Specify a valid drive.

**Invalid parameter**

One of the switches that you have specified is wrong.

**Invalid sub-directory entry**

The subdirectory that you have specified either does not exist or is invalid. Check to see that you have entered the subdirectory name correctly.



Invalid working directory

Processing cannot continue Your disk is bad. Replace the disk or make a copy from your backup system disk.

Probable non-DOS disk Continue (Y/N)?

The disk you are using is a non-DOS disk. You must indicate whether or not you want CHKDSK to continue processing. This error message usually means that the File Allocation Table (FAT) is bad.

Unrecoverable error in directory

Convert directory to file (Y/N)?

If you respond Y to this prompt, CHKDSK converts the bad directory into a file. You can then delete it.

If you respond N to this prompt, you may not be able to write to or read from the bad directory.

## CLS (Clear screen)

---

**Purpose:** Clears the terminal screen.

**Type:** Internal

**Format:** CLS

**Remarks:**

The CLS command clears the screen and leaves the cursor in the upper, left corner.

## COMMAND

---

**Purpose:** Starts the command processor.

**Type:** External

**Format:** COMMAND [d:][path] [cttydev] [/P]  
[/E:nnnnn] [/C string]

where:

*d: path* tells the command processor where to look for the COMMAND.COM file if it needs to reload the transient part into memory.

*cttydev* allows you to specify a different device (such as AUX) for input and output. See the CTTY command for more information.

**Switches:**

/E specifies the environment size, where *nnnnn* is the size in bytes. The size may range between 128 and 32768 bytes. The default value is 128 bytes.

/P tells COMMAND.COM not to exit to any higher level.

/C tells the command processor to execute the command or commands specified by *string* and then return. If used, should be the last switch in the command.

**Remarks:**

This command starts a new command processor (the MS-DOS program that contains all internal commands).

The command processor is loaded into memory in two parts: the transient and the resident. Some application programs write over the transient part of COMMAND.COM when they run. When this happens, the resident part of the command processor looks for the COMMAND.COM file on disk so it can reload the transient part.

If *nnnnn* is less than 128 bytes, MS-DOS defaults to 128 bytes and gives the message:

INVALID ENVIRONMENT SIZE SPECIFIED

If *nnnnn* is greater than 32768 bytes, MS-DOS gives the same message, but defaults to 32768 bytes.

**Example:**

COMMAND /C CHKDSK B:

This example tells the command processor to:

1. Start a new command processor under the current program.
2. Run the command CHKDSK B:.
3. Return to the first command processor.

## COMP (Compare)

---

**Purpose:** Compares the contents of two files.

**Type:** External

**Format:** COMP [d:][path][filename1[.ext]]  
[d:][path][filename2[.ext]]

where:

*d:path filename1.ext* is the first file.

*d:path filename2.ext* is the file you want to compare with the first.

### Remarks:

If you do not specify a disk drive, COMP uses the default drive.

If you do not specify a file name, COMP uses \*.\*.

If you specify only a disk drive for the second file, COMP assumes the second file name is the same as the first file name.

Use global file names to compare files with same names but different extensions.

For example, the following command instructs COMP to compare each .BAS file in the current directory on drive C: with a file of the same name, but extension .BAK on drive A:.

```
COMP C:* .BAS A:* .BAK
```

Use the COMP command with only paths to compare all files in one directory with the corresponding files in another directory.

Use the COMP command with no parameters to compare files on different diskettes. When COMP prompts for parameters, insert the appropriate diskette and answer the prompt by typing the file names you want to compare.

See the DISKCOMP utility in this manual for information on comparison of diskette contents.

Files must be the same size.

If the comparison is successful, COMP displays the following message:

```
FILES COMPARE OK
```

COMP then prompts :

```
COMPARE MORE FILES (Y/N)?
```

To compare more file, type Y. COMP then prompts for more file names.

To exit COMP type N.

### Examples:

```
COMP A:* .BAS B:* .BAK
```

Compares each file with extension .BAS on drive A: with a file of the same name, but extension .BAK on drive B:.

```
COMP A:* .COM C:
```

Compares all .COM files on drive A with the files of the same name and extension on drive C:.

```
COMP C:\SUBDIR1 C:\SUBDIR2
```

Locates and compares all files in a subdirectory SUBDIR1 with files of the same name in SUBDIR2 on drive C.

**Messages:**

File not found

You typed incorrect file name.

Compare error at offset xxxxxx

File 1 = xx

File 2 = xx

Files contain mismatched data.

Compare error at offset xxxxxx

Indicates the offset into the files of the mismatching bytes.

File 1 = x, File 2 = x

Indicates the contents of the mismatching bytes in both files. COMP stops processing after the mismatches and displays the following message:

10 Mismatches - ending compare.

EOF mark not found

COMP cannot find an End-of-File mark (EOF).



## COPY

---

**Purpose:** Copies one or more files to another disk, combines and appends files on the same disk, copies screen input to a file.

**Type:** Internal

**Format:**

*To copy:*

COPY source[/A][/B] target[/A][/B][/V]

where:

*source* is [d:][path] [filename1[.ext]].

*target* is [d:][path] [filename2[.ext]].

*To combine or append:*

COPY source1[/A][/B] + [source2[/A][/B]...]  
target[/A][/B][/V]

where:

*source1* is [d:][path] [filename1[.ext]].

*source2* is [d:][path] [filename2[.ext]].

*target* is [d:][path] [filename3[.ext]] and is a combination of all previous files in the command.

*To copy from screen input to a file:*

**COPY CON filename**

where *filename* is the target file.

**Switches:**

**/A** Indicates the files being processed are ASCII files. /A applies to the file name preceding it, and to all remaining file names in the command, until another /A or /B is encountered.

When /A is used with a *source file* data in the file is copied up to but not including the first end-of-file mark (in EDLIN, this is Control-Z). The remainder of the file is not copied.

When used with a *target file*, /A causes an end-of-file character to be added as the last character of the file.

When you are appending files, the default switch is /A.

**/B** indicates the files being processed are binary files. /B applies to the file name preceding it, and to all remaining file names in the command, until another /A or /B is encountered.

When used with a *source filename*, /B causes the entire file to be copied, including any end-of-file mark.

When used with a *target filename*, /B causes no end-of-file character to be added.

**/V** verifies the sectors written on the target disk are recorded properly. Although recording errors are rare, you can verify data has been correctly recorded. /V causes the COPY command to run slower because MS-DOS must check each entry recorded on the disk. An error is displayed if a write is not verified.

**Remarks:***When copying files:*

If the source and target files are both in the working directory, you only need to use file names.

The target may take these forms:

1. If the target is a drive name only, the source file is copied with the source file name to the designated drive.
2. If the target is a file name only, the source file is copied to a file on the default drive with the file name specified.
3. If the target includes both a drive and file name, the original file is copied to a file on the drive specified.
4. If the target file does not exist, MS-DOS creates it.
5. If the target file does exist, it will be overwritten by the source file.
6. If the target is not specified and the source file resides on another drive, the source file with the source file name is copied to the current drive.

*When combining or appending files:*

If no target is specified, the source files are appended to the first file.

### Examples:

#### *To copy files:*

The following command will make a copy on drive B: of MEMO.DOC named MEMO.DOC.:

```
COPY MEMO.DOC B:
```

The following command will make a copy of MEMO.DOC, name it LETTER.DOC, and place it on the default drive:

```
COPY MEMO.DOC LETTER.DOC
```

The following command will make a copy of MEMO.DOC on the default drive, name the copy MEMO.DOC, and place the copy on the disk in drive B:.

```
COPY MEMO.DOC B:MEMO.DOC
```

The following command copies all of REPORT.EXE including the end-of-file mark to REPORT2.EXE:

```
COPY REPORT.EXE/B REPORT2.EXE
```

The following command, copies MEMO.TXT to LETTER.TXT and adds an end-of-file character to LETTER.TXT:

```
COPY MEM.TXT LETTER.TXT/A
```

*To combine and append files:*

To combine files named INTRO.RPT, BODY.RPT, and SUM.RPT (on drive B) and place them in the file on the default drive called REPORT, type:

```
COPY INTRO.RPT + BODY.RPT + B:SUM.RPT REPORT
```

To combine all files with a file name extension of .LST into a file named COMBIN.PRN, type:

```
COPY *.LST COMBIN.PRN
```

To combine files with the same file names but different file name extensions (.LST and .REF) into COMBIN.TEX, type:

```
COPY *.LST + *.REF COMBIN.TEX
```

To create files by combining files with the same file names but different file name extensions, type:

```
COPY *.LST + *.REF *.PRN
```

In the preceding example, each .LST file is combined with the corresponding .REF file. The result is a file with the same file name but with the extension .PRN. So, FILE1.LST will be combined with FILE1.REF to form FILE1.PRN; then FILE2.LST with FILE2.REF to form FILE2.PRN; and so on.

To append FILES2 and FILES3 to FILES1, type:

```
COPY FILES1 + FILES2 + FILES3
```

COPY compares the source file name to the target file name when files are combined. If they are the same, the source file is skipped, an error message is displayed, and COPY continues with the remaining files in the command. For example:

```
COPY X + Y + Z Y
```

COPY displays an error message when it tries to copy source file Y into target file Y.

*To copy screen input to a file:*

To use COPY to put your screen input into MYFILE, enter:

```
COPY CON MYFILE <Enter>
```

```
DEAR MR. DIMM, <Enter>
```

```
WE WOULD LIKE TO INFORM YOU <Enter>
```

```
<Control>-<Z> <Enter>
```

After you press <Control>-<Z> and <Enter>, your text will be saved in MYFILE.

**Messages:**

Cannot do binary reads from a device

The copy cannot be done in binary mode when you are copying from a device. Remove the /B switch or specify an ASCII copy with the /A switch.

Content of destination lost before copy

A file to be used as a source file has been overwritten prior to completion of the copy.

File cannot be copied onto itself

0 File(s) copied

If the first path or file is one the default drive and you do not specify a second path, COPY will quit. Copying files to themselves is not allowed.



## CTTY (Change console)

---

**Purpose:** Allows you to change the device from which you issue commands. (In this command, the letters TTY represent the console; that is, your keyboard.)

**Type:** Internal

**Format:** CTTY DEVICE

where *device* is the device from which you are giving commands.

**Remarks:**

CTTY is useful if you want to change the device on which you are working.

The command CTTY AUX moves all command I/O (input/output) from the current device (the console) to an AUX port such as another terminal. The command CTTY CON moves I/O back to the console. Refer to Section 1.3 for a list of valid device names to use with the CTTY command.

### NOTE

There are many programs that do not use MS-DOS for input, output, or both. These programs do input directly to the hardware on your computer. The CTTY command will have no effect on these programs. CTTY will only affect programs that use MS-DOS.

## DATE

---

**Purpose:** Enter or change the date known to the system. This date will be recorded in the directory for any files you create or change.

**Type:** Internal

**Format:** DATE [mm-dd-yy]

where *mm-dd-yy* specifies the month, date, and year.

**Remarks:**

You can change the date from your terminal or from a batch file. (MS-DOS does not display a prompt for the date if you use an AUTOEXEC.BAT file, so you may want to include a DATE command in that file.)

If you type:

DATE

DATE responds with the message:

CURRENT DATE IS (WEEKDAY) (MM)-(DD)-(YY)  
ENTER NEW DATE (MM-DD-YY):

Press <Enter> if you do not want to change the date shown.

You can also type a particular date after the DATE command, as in:

DATE 3-9-86

In this case, the ENTER NEW DATE: prompt does not appear after you have pressed <Enter>.

Use only numbers when you type the date; allowed numbers are:

*mm* = 1-12

*dd* = 1-31

*yy* = 80-99 (or 1980-2099)

The date, month, and year entries may be separated by hyphens (-) or slashes (/). MS-DOS is programmed to change months and years correctly, whether the month has 31, 30, 29, or 28 days. MS-DOS compensates for leap years, too.

It is possible to change the mm-dd-yy format in which the date is displayed and entered. The COUNTRY command in the CONFIG.SYS file allows you to change the date format to the European standard dd-mm-yy.

### Messages:

Invalid date. Enter new date (mm-dd-yy):

If the numbers or separators are not valid, DATE displays this message. Enter a valid date.

## DEL (Delete)

---

**Synonym:** ERASE

**Purpose:** Deletes all files with the designated file specification.

**Caution:** If you enter DEL path and reply Y to "Are you sure?", *all* files in the directory specified will be deleted.

If the filename.ext is \*.\* , the prompt "Are you sure?" appears. If a Y is entered, *all* files on the current directory will be deleted.

**Type:** Internal

**Format:** DEL [d:] [path][filename[.ext]]

where *d: path filename.ext* are the files to delete.

If *d:* is not specified, the default drive is assumed.

If *path* is not specified, the current directory is assumed.

If *filename* is not specified, all of the files in the *path* are deleted.

**Remarks:**

You cannot DELETE system or read-only files.

You cannot DELETE a directory. You can use DELETE to erase the files within the directory, but you must use the RMDIR command to delete a directory.

## DIR (Directory)

---

**Purpose:** List the files in a directory.

**Type:** Internal

**Format:** DIR [d:][path][/P][/W]

where *d: path* specifies which directory to list.

**Switches:**

/P causes the screen display to pause after the screen is filled. To resume display of output, press any key.

/W selects wide display. Only file names are displayed, without other file information. Files are displayed five per line.

**Remarks:**

If you type DIR, all directory entries on the default drive are listed. If you include a drive name, such as DIR B:, all entries on the disk in the specified drive are listed. If only a file name is entered with no extension (DIR FILENAME), then all files with the designated file name on the disk in the default drive are listed.

When you type a file name with a drive letter (for example, DIR B:FILENAME.EXT), all files with the file name specified on the disk in the drive specified are displayed. In all cases, files are listed with their size in bytes and with the time and date of their last modification. The wildcards ? and \* (question mark and asterisk) may be used in the file name option.

The following DIR commands are equivalent:

DIR

DIR \*.\*

DIR filename

DIR filename.\*

DIR .ext

DIR \*.ext

If the COUNTRY command in the CONFIG.SYS file is set to a country other than the U.S., the directory date and time formats will differ.

#### NOTE

The DIR command cannot list the directory if you specify a path name of more than 64 characters (including the drive name). Therefore, you may need to change directories if you want to list files in deep subdirectories.

## DISKCOMP (Disk compare)

---

**Purpose:** Compares the contents of the disk in the source drive to the disk in the target drive.

**Type:** External

**Format:** DISKCOMP [d1:] [d2:] [/1] [/8]

where:

*d1*: is the drive containing the source diskette.

*d2*: is the drive containing the target diskette.

**Switches:**

- /1 compares only the first side of the disk, even if the disks and drives you are using are double-sided.
- /8 compares only the first 8 sectors per track, even if the disks contain 9 or 15 sectors per track.

**Remarks:**

If you specify only one drive, DISKCOMP uses the default drive as the target drive. If you specify the same drive as the source and target, DISKCOMP does a comparison using one drive, and prompts you to insert the disks as appropriate.

DISKCOMP performs a track-by-track comparison of the disks. It automatically determines the number of sides and sectors per track based on the format of the source disk. If the target disk is not the same type as the disk in the source drive, DISKCOMP displays the message:

Drive types (double, single-sided) or diskette types not  
compatilble

If all of the tracks are the same, DISKCOMP displays the message:

Diskettes compare OK

If the tracks are not the same, DISKCOMP displays a "Compare error" message that includes the track and side number where it found the mismatch.

When DISKCOMP completes the comparison, it prompts:

Compare more diskettes (Y/N)?

If you type Y, DISKCOMP prompts you to insert the proper disks and does the next comparison. If you type N, DISKCOMP ends. If you end DISKCOMP and if the disk in the default drive does not contain MS-DOS, DISKCOMP prompts you:

Insert disk with COMMAND.COM in drive A and strike any  
key when ready



You cannot use DISKCOMP with assigned, joined, or substituted drives. DISKCOMP does not work on network drives. If you attempt to use the DISKCOMP command with these types of drives, DISKCOMP displays an error message.

### NOTE

If you are comparing a disk with a backup disk that you made with the COPY command, DISKCOMP may give the "Compare error" message, even if the files on the disks are identical. This is because the COPY command duplicates the information, but it doesn't necessarily place the information in the same location on the target disk. In this case, you should use the FC utility to compare individual files on the disk.

When correctly written programs exit back to DOS, they return an error code: 0 if no error occurred, or a value greater than zero if there was a problem. This error code can be tested in batch files, and it allows batch programmers to branch to an error-handling routine in the batch file.

The DISKCOMP command returns the following error level codes:

- 0 Compared OK The disks compared exactly.
- 1 Did not compare The disks were not the same.
- 2 Control-C error The user terminated with Control-C.
- 3 Hard error An unrecoverable read or write error occurred --did not compare.
- 4 Initialization error There is not enough memory--invalid drives or command line syntax.

**Messages:**

- Cannot DISKCOMP to or from a network drive  
One of the drives that you specified is a network device.
- Cannot DISKCOMP to or from an ASSIGNED or SUBSTed drive  
One of the drives that you specified is a drive that you created using the ASSIGN or SUBST command.
- Compare another diskette (Y/N)?  
After comparing the disks, DISKCOMP asks if you want to compare another disk. Press Y (for Yes) or N (for N).
- Compare error on side <NNN> , track <nnn>  
DISKCOMP found an error on side *NNN*, track *nnn*.
- Compare OK  
DISKCOMP displays this message if the disks are identical.
- Compare process ended  
DISKCOMP displays this message if a fatal error occurred during the comparison.
- Comparing (tt) tracks (xx) sectors per track, (y) side(s)  
This message gives the format (of the first disk) that DISKCOMP is using to perform the comparison.

DEVICE Support Not Present

The disk drive does not support MS-DOS 3.x device control.

Do not specify filename(s) Command format: DISKCOMP d:  
d:[/1][/8]

You specified an incorrect switch or gave a file name in addition to a drive name.

Drive (x:) not ready

Make sure a diskette is inserted into the drive and the door is closed. You have not inserted a disk into the specified disk drive.

Drive types or diskette types not compatible

You cannot compare a single-sided disk with a double-sided disk, or a high-density disk with a low-density disk. You should use FC to compare the files on the disks.

FIRST diskette bad or incompatible DISKCOMP cannot determine the format of the disk that you want to compare.

Incorrect DOS version. You must use the correct version of MS-DOS with this command.

Insert FIRST diskette in drive (x:)

Press any key when ready . . .

DISKCOMP displays this message to prompt you to insert the first disk into the specified drive.

Insert SECOND diskette in drive (x:)

Press any key when ready . . .

DISKCOMP displays this message to prompt you to insert the second disk into the specified drive.

**Insufficient memory**

There is not enough memory available to perform the comparison.

**Invalid drive specification**

You specified an drive that is incorrect or doesn't exist.

**Invalid parameter**

You specified a parameter that is incorrect or doesn't exist.

**SECOND diskette bad or incompatible**

The second disk does not contain the same format as the first disk, or DISKCOMP does not recognize the format of the second disk.

**Specified drive does not exist or is non-removable**

You must specify the name of a valid floppy drive.  
DISKCOMP cannot compare hard disks.

**Unrecoverable read error on drive (x:)**

DISKCOMP could not read the disk.

## DISKCOPY

---

**Purpose:** Copies the contents of the disk in the source drive to the disk in the target drive.

**Type:** External

**Format:** DISKCOPY [S:] [T:] /1

where:

S: is the source drive.

T: is the target drive.

**Switches:**

/1 Copy only the first side of the diskette even if the drive is double sided.

**Remarks:**

If the disk in the target drive is unformatted, DISKCOPY formats it with the same format that is on the source disk.

You can specify the same drives or you may specify different drives. If the drives are the same, a single-drive copy operation is performed. You are prompted to insert the disks at the appropriate times. DISKCOPY waits for you to press any key before continuing.

After copying, DISKCOPY prompts:

Copy complete Copy another (Y/N)?

If you press Y, MS-DOS will prompt you to insert source and target disks, and the next copy is performed on the same drives that you originally specified.

To end the DISKCOPY process, press N.

If you omit both options, a single-drive copy operation is performed on the default drive. If you omit the second option, the default drive is used as the target drive.

Both disks must have the same number of physical sectors and those sectors must be the same size.

Disks that have had a lot of files created and deleted on them become fragmented, because disk space is not allocated sequentially. (The first free sector found is the next sector allocated, regardless of its location on the disk.) A fragmented disk can cause slowness due to delays in finding, reading, or writing a file. If this is the case, *do not* use DISKCOPY to copy the diskette. Use either the COPY command or the XCOPY command to copy the desired files to a *newly formatted diskette*. (If the diskette is not *newly formatted*, fragmented files may still be produced.) COPY and XCOPY will write the files in sequential order.

## Error Messages:

When correctly written programs exit back to DOS, they return an error code: 0 if no error occurred, or a value greater than zero if there was a problem. This error code can be tested in batch files, and it allows batch programmers to branch to an error-handling routine in the batch file.

- |   |  |
|---|--|
| 0 | Copied Successfully<br>The last DISKCOPY was completed with no errors.                     |
| 1 | Non-fatal read/write error<br>An unrecoverable but non-fatal read or write error occurred. |
| 2 | Control-C error<br>The user entered Control-C to terminate DISKCOPY.                       |
| 3 | Fatal hard error<br>DISKCOPY was unable to read the source disk or format the target disk. |
| 4 | Initialization error<br>There is not enough memory--invalid drives or command line syntax. |

**Messages:**

Cannot DISKCOPY to or from a network drive

The user is trying to DISKCOPY to or from a network drive.

Cannot DISKCOPY to or from an ASSIGNED or SUBSTed drive

The user is trying to DISKCOPY to or from a "virtual" drive that was created by using the SUBST or ASSIGN command.

Copy another diskette (Y/N)?

The user prompted to see if the DISKCOPY process was to be repeated on another source/target disk.

Copy not completed

DISKCOPY cannot copy the entire disk. Use the COPY command to copy specific files onto another disk.

Copying NN tracks NN Sectors/Track, NN Side(s)

DISKCOPY describes the format of the SOURCE disk.

Disk error while reading drive A: Abort, Ignore, Retry?

DISKCOPY found errors during processing.

Disks must be the same size

You cannot copy the contents of a source disk with a format different from the target disk using DISKCOPY. Use the COPY command to copy files onto the disk.



Drive (x:) not ready

Make sure a diskette is inserted into the drive and the door is closed. The disk is not ready to be read or written to.

Drive types or diskette types not compatible

For example: Attempting to copy to a high-density disk in a low-density drive.

Formatting while copying

If the target disk is not the same format as the source disk, DISKCOPY reformats the target disk.

Incorrect DOS version

Many version 2.x and 3.x utilities will not run on earlier versions of MS-DOS. Some utilities will only run under the exact version of MS-DOS for which they were configured.

Insert [SOURCE|TARGET] diskette in drive D:

Press any key when ready...

This is the prompt to begin the DISKCOPY process.

Insufficient memory

There is not enough memory to perform the DISKCOPY.

Invalid drive specification

User specified drive that does not exist or is non-removable.

Invalid parameter, Do not specify filename(s)

Command Format: DISKCOPY (x:) [/1]

The user specified either an illegal switch or file names  
instead of only drive names.

[SOURCE | TARGET] diskette bad or incompatible

Copy process ended

DISKCOPY could not read the source diskette, or the  
target diskette could not be read or formatted.

Target diskette is write protected

The target disk has a write protect tab on it.

Target diskette may be unusable

Unrecoverable read or write errors were encountered  
while copying to the target disk. The target disk is not an  
exact copy of the source disk.

Unrecoverable [read | write] error on drive (x:) side NN, track NN

An error reading/writing to the disk occurred.

Source and target diskettes are not the same format.

Cannot do the copy

You must have the same size and kind of disks to run  
DISKCOPY. For example, you cannot copy from a  
single-sided disk to a double-sided disk. Reformat the  
target disk to be of the same type as the source disk or  
substitute a disk of the same type.

## EXE2BIN (Executable to binary)

---

**Purpose:** Converts .EXE (executable) files to binary format.

**Type:** External

**Format:** EXE2BIN [d:][path] filename1.ext  
[d:][[path] filename2.ext]

where:

*d:path filename1.ext* is the input file.

*d:path filename2.ext* is the output file

### Remarks:

This command is useful only if you want to convert .EXE (executable) files to binary format. The file named by *path name* is the input file. If no extension is specified, it defaults to .EXE. The input file is converted to .BIN file format (memory image of the program) and placed in the output file (second path name).

If you do not specify a drive name, the drive of the input file will be used. If you do not specify an output file name, the input file name will be used. If you do not specify a file name extension in the output file name, the new file will be given an extension of .BIN.

The input file must be in valid .EXE format produced by the linker. The resident, or actual code and data part of the file must be less than 64K. There must be no STACK segment.

Two kinds of conversions are possible, depending on whether the initial CS:IP (Code Segment: Instruction Pointer) is specified in the .EXE file:

1. If CS:IP is not specified in the .EXE file, a pure binary conversion is assumed. If segment fixups are necessary (that is, the program contains instructions requiring segment relocation), you will be prompted for the fixup value.

This value is the absolute segment at which the program is to be loaded. The resultant program will be usable only when loaded at the absolute memory address specified by a user application. The command processor will not be able to load the program.

2. If CS:IP is 00:100H, it is assumed that the file will run as a .COM file with the location pointer set at 100H by the assembler statement ORG; the first 100H bytes of the file are deleted. No segment fixups are allowed, as .COM files must be segment relocatable; that is, they must assume the entry conditions explained in the *Microsoft Macro Assembler Manual*.

Once the conversion is complete, you may rename the output file with a .COM extension. Then the command processor will be able to load and execute the program in the same way as the .COM programs supplied on your MS-DOS disk.

## Messages:

### File cannot be converted

CS:IP does not meet either of the criteria specified above, or it meets the .COM file criterion but has segment fixups. This message is also displayed if the file is not a valid executable file.

### File not found

The file is not on the disk specified.

### Insufficient memory

There is not enough memory to run EXE2BIN.

### File creation error

EXE2BIN cannot create the output file. Run CHKDSK to determine if the directory is full, or if some other condition caused the error.

### Insufficient disk space

There is not enough disk space to create a new file.

### Fixups needed base segment (hex):

The source (.EXE) file contained information indicating that a load segment is required for the file. Specify the absolute segment address at which the finished module is to be located.

### File cannot be converted

The input file is not in the correct format.

### WARNING Read error in EXE file.

#### Amount read less than size in header

This is a warning message only. It means the .EXE header in the file is inconsistent with the size of the file.

## EXIT

---

**Purpose:** Exits the program COMMAND.COM (the command processor) and returns to a previous level, if one exists.

**Format:** EXIT

**Type:** Internal

**Remarks:**

This command can be used when you are running an application program and want to start the MS-DOS command processor, then return to your program.

For example, to look at a directory on drive B: while running an application program, you must issue an EXEC of the command interpreter (system call 4BH). The system prompt will appear. You can now type the DIR command and MS-DOS will display the directory listing. When you type EXIT, you return to the previous level (your application program).

## FDISK

---

**Purpose:** Prepares your hard disk to work with MS-DOS by creating DOS partitions.

**Caution:** *Using different versions of DOS after preparing your disk with the FDISK command provided on your system's MS-DOS diskette could cause loss of data.*

The FDISK command provided on your MS-DOS diskette is an enhanced version of the standard FDISK command. The ability to create and use multiple DOS partitions is available *only* by using the the MS-DOS diskette provided with your system. Other versions will not recognize multiple partitions.

**Type:** External

**Format:** FDISK

**Remarks:**

If your disk is 32 megabytes (MB) or less and you plan to use only MS-DOS, you only need one partition.

If your disk is larger than 32 megabytes (MB), you will need multiple partitions. MS-DOS can only address/recognize 32 MB or less of disk space per drive or partition. Therefore, if your disk is larger than 32 MB you must have multiple partitions. For example, if you had a 70 MB fixed disk, you could create three partitions. The first partition contains 30 MB of disk space, the second 20 MB, and the third 20 MB. MS-DOS handles each partition as a separate drive and will assign each partition a drive letter when the system is booted.

If you wish to use multiple operating systems on a single disk, you should create multiple partitions. If you have multiple operating systems, each operating system must reside in separate partition. So, if you wanted to use both MS-DOS and XENIX® you would create two partitions. One partition would contain MS-DOS and be created by this FDISK command. (The DOS partition should not take the entire disk.) The other partition would contain XENIX and be created by a XENIX command similar to FDISK.

FDISK can create a maximum of four DOS partitions on a single, physical disk.

After running FDISK, you must run FORMAT on the newly-created DOS partitions. (FDISK creates partitions; FORMAT creates control structures within the partition that allow MS-DOS to store data.)

FDISK also allows you to perform the following hard disk tasks:

- Change an active partition.
- Delete a partition.
- Display partition data.
- Select the next hard disk drive for partitioning, if you have more than one hard disk.



After you enter the FDISK command, the following Option Menu displays:

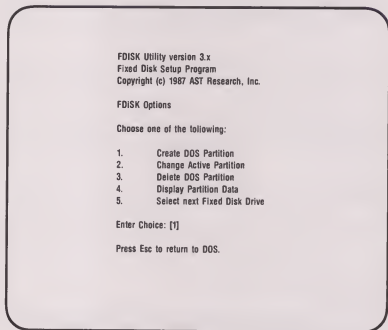
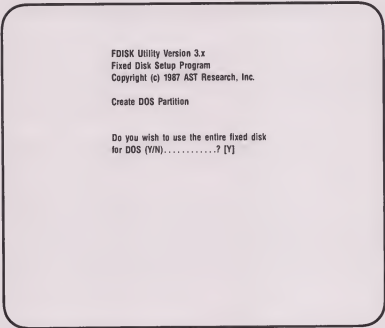


Figure 3-1. FDISK Option Menu.

### Option 1: Creating a DOS Partition

Option 1 is used to create a DOS partition. Up to four DOS partitions can be created on a single hard disk.

To create a DOS partition, type **1** in the Options Menu and press **<Enter>**. A screen similar to the following example appears:



```
FDISK Utility Version 3.x
Fixed Disk Setup Program
Copyright (c) 1987 AST Research, Inc.
```

Create DOS Partition

Do you wish to use the entire fixed disk  
for DOS (Y/N).....? [Y]

Figure 3-2. Creating a DOS Partition.

If your fixed disk is smaller than 32 MB, the system asks:

Do you wish to use the entire fixed disk for DOS (Y/N)?

If you will only be using DOS, type Y and press < Enter > . The following message appears:

System will now restart  
Insert DOS diskette in Drive A:  
Press any key when ready...

Insert an MS-DOS diskette in drive A: and press any key. DOS restarts and prompts you to enter the date and time.

If your disk is larger than 32 MB or you enter N at the "Do you wish to use entire fixed disk?" question, FDISK displays the following message:

Total disk space is xxxx cylinders

Maximum available space is xxxx cylinders at xxxx

*Total disk space* indicates the number of cylinders on your disk.

*Maximum available space* indicates the maximum number of cylinders you can use in this partition. The maximum available space is either the number of cylinders equal to 32 MB or the number of free cylinders remaining, whichever is less.

After displaying information on disk space, FDISK prompts:

Enter Partition Size: [xxxx]

Enter the partition size you want in cylinders. The following prompt appears:

Enter starting cylinder number: [xxxx]

where xxxx is the default cylinder number.

To locate the DOS partition at the default cylinder number, press <Enter> . To locate the partition elsewhere, type in the starting cylinder number and press <Enter> .

When the partition is complete, the following message displays:

Press ESC to return to Options Menu

Press <Esc> to return to the Options menu. Select Option 1: Creating a DOS Partition and repeat this procedure until you have created the desired number of partitions.

#### NOTE

If a partition starts on a bad spot on your disk, FDISK automatically adjusts the starting point to the first good area on the disk. This adjustment may reduce the size of your partition.

## **Option 2: Changing the Active Partition**

Option 2 permits you to change the active partition on your hard disk. The system will boot from the active partition. Only one partition can be active on a hard disk. If you have multiple DOS partitions, only the first DOS partition should be active. (If you have more than one disk only the first DOS partition on the first disk should be active.)

For example, assume your system is divided into two DOS partitions and one XENIX partition. You would use this option to designate the first DOS partition as active.

To change the active partition, type 2 on the FDISK Options Menu. FDISK displays the partition status of your hard disk.

The following sample screen shows a hard disk with three partitions:

FDISK Utility version 3.x  
Fixed Disk Setup Program  
Copyright (c) 1987 AST Research, Inc.

Change Active Partition

Partition	Status	Type	Start	End	Size
1	A	DOS	000	149	150
2	N	DOS	150	304	155
3	N	non-DOS	305	449	150

Total disk space is 455 Cylinders

Enter the number of the partition you  
want to make active: [ ]

Figure 3-3. Changing the Active Partition.

The Change Active Partition screen columns provide the following information:

- *Partition* lists the number assigned to each partition.
- *Status* lists the partition status. A for Active and N for non-active.
- *Type* list partition classification, DOS or non-DOS.
- *Start and End* list the cylinder numbers of the partition.
- *Size* is the number of cylinders the partition uses.

To change the active partition, type the partition number that you want to activate and press <Enter>.

### Option 3: Deleting a DOS Partition

Option 3 permits you to erase all data within a DOS partition and remove the partition boundaries.

To select a DOS partition to delete, type 3 on the FDISK Options Menu.

A screen similar to the following appears:

```
FDISK Utility Version 3.x
Fixed Disk Setup Program
Copyright (c) 1987 AST Research, Inc.
```

```
Create DOS Partition
```

Partition	Status	Type	Start	End	Size
1	A	DOS	000	140	150
2	N	DOS	150	304	155

```
Total disk space is 455 Cylinders
The current active partition is 1
```

```
Which partition do you wish to delete.....? [1]
```

```
Warning! Data in the DOS partition
could be lost. Do you wish to
continue.....? [N].
```

Figure 3-3. Deleting a DOS Partition.



MS-DOS displays the following question:

Which partition do you wish to delete?

Only DOS partitions can be deleted. Enter the number of the appropriate partition. MS-DOS asks:

Warning! Data in the partition could be lost. Do you wish to continue? [N]

Respond to the prompt as follows:

- To cancel the deletion, press < Enter > and FDISK returns you to the Options Menu. [N] is the default.
- To delete the DOS partition, type Y and press < Enter > .

### Option 4: Displaying Partition Information

Option 4 lists status, type, and size of all partitions.

Type 4 on the Options Menu and a screen similar to the following appears:

```
FDISK Utility Version 3.x  
Fixed Disk Setup Program  
Copyright (c) 1987 AST Research, Inc.
```

#### Display Partition Information

Partition	Status	Type	Start	End	Size
1	A	DOS	000	149	150
2	N	DOS	150	304	155
3	N	non-DOS	305	449	150

Total disk space is 450 Cylinders

Figure 3-4. Displaying Partition Information.

### **Option 5: Select the Next Hard Disk Drive**

This option displays only if your system has more than one hard disk.

If you have two hard disks, type **5** in the Options Menu to partition the second disk.

After you have entered the option, you see the current hard disk drive number change on the FDISK options menu.

## FIND

---

**Purpose:** Searches for a specific string of text in a file or files.

**Type:** External

**Format:** `FIND [/V] [/C] [/N] "string" [d:]  
[path][filename[.ext]]`

where:

*"string"* is the text to be searched for. The string must be enclosed by quotation marks.

*d: path filename.ext* is the drive, path, and file to be searched.

**Switches:**

- `/V` displays all lines not containing the specified string.
- `/C` prints only the count of lines that contained a match in each of the files.
- `/N` precede each line by its relative line number in the file.

**Remarks:**

FIND is a filter that takes as options a string and a series of file names. It will display all lines that contain a specified string (group of characters) from the files in the command line.

If no file names are typed, FIND takes the input from the screen and displays all lines containing the specified string.

**Examples:**

```
FIND "FOOL'S PARADISE" BOOK1.TXT BOOK2.TXT
```

This command displays all lines from BOOK1.TXT and BOOK2.TXT (in that order) that contain the string "Fool's Paradise."

The command

```
DIR B: | FIND /V "DATE"
```

causes MS-DOS to display all names of the files on the disk in drive B: that do not contain the string "DATE". Type double quotes around a string that has quotes in it.

**Messages:****Incorrect DOS version**

Find will only run on versions of MS-DOS that are 2.0 or higher.

**Find: Invalid number of parameters**

You did not specify a string when issuing the FIND command.

**Find: Format error**

You typed an illegal string when issuing the FIND command. Remember that the string must be enclosed by quotation marks.

**Find: File not found <filename>**

The file name you have specified does not exist or FIND cannot find it.

**Find: Read error in <filename>**

An error occurred when FIND tried to read the file specified in the command.

**Find: Invalid parameter <option-name>**

You specified an option that does not exist.

## FORMAT

---

**Purpose:** Prepares a disk to accept MS-DOS files.

**Type:** External

**Caution:** FORMAT destroys all data on the disk.

**Format:** `FORMAT d: [/S] [/1] [/8] [/V] [/4] [/B]`

where: *d*: Is the drive containing the disk you want to format. If you do not specify *d*, FORMAT uses the default drive.

**Switches:**

- `/S` copies the operating system files to the diskette.
- `/1` formats a diskette for single-sided use only.
- `/8` formats a diskette for 8-sector use. Do not use /8 with /B.
- `/V` prompts for a volume name to write to the disk. The name can be up to 11 characters long. You can use any character that is valid for a filename. Do not use /V with /B.
- `/4` formats a low-capacity disk in a high-capacity drive.
- `/B` reserves areas of the disk for subsequent addition of system files.

Only use switches compatible with your disk or diskette. The following table lists the switches you can use with specific disks or diskettes.

Disk Type	Compatible Switches
160/180 KB	/S, /V, /1, /8, /B, /4
320/360 KB	/S, /V, /8, /B, /4
720 KB	/S, /V
1.2 MB	/S, /V
1.44 MB	/4, /B, /V, /S
Hard Disk	/S, /V

**Remarks:**

FORMAT should be run on each MS-DOS partition created by FDISK.

Do not use FORMAT to prepare disks on drives you have reassigned with the ASSIGN or SUBST commands. Substituted drives can hide the real drive letters FORMAT uses.

Do not use FORMAT while a JOIN command is in effect. JOIN connects drive letters to path names and can invalidate drive names.

Do not use FORMAT on network drives.



FORMAT sets the following error codes:

- 0 Successful completion of the most recent format.
- 3 You have ended a format with <Ctrl> - <Break>

(Error codes 1 and 2 are undefined.)

**Example:**

The following command formats the diskette in drive B: and copies MS-DOS system files to the diskette.

```
FORMAT B:/S
```

**Messages:**

Enter current Volume Label for Drive:

Displays when you try to format a hard drive containing data. Enter the drive's volume label. If no volume label was assigned, press <Enter>.

Format complete

FORMAT displays this message when it has finished formatting the disk in the specified drive.

Head: (hh) Cylinder: (cc)

FORMAT displays the current head and cylinder number being formatted.

Insert new diskette for drive x:  
and strike ENTER when ready

Insert the diskette to be formatted in drive A:. Press <Enter> to begin the format. If there is any data on the disk, FORMAT will destroy it. Press <Ctrl>-<C> if you don't want to format the disk.

System transferred

The system files MSDOS.SYS and IO.SYS have been transferred to the disk being formatted.

Volume label (11 characters, ENTER for none)?

This message appears when you use the /V switch. Enter a label of up to 11 characters to identify the disk, or press <Enter> if no label is needed.

**WARNING, ALL DATA ON NON-REMOVABLE DISK  
DRIVE x: WILL BE LOST!**

Proceed with Format (Y/N)?

MS-DOS is confirming you want to proceed. If you proceed with FORMAT all the data on the disk will be destroyed. To continue, press Y (for yes). If you do not want the files or your hard disk erased, press N (for no). Copy the files onto floppy disks and repeat the FORMAT command.

## GRAFTABL (Graphics table)

---

**Purpose:** Loads additional character data into a table in memory for use with a color or graphics adapter.

**Type:** External

**Format:** GRAFTABL

**Remarks:**

GRAFTABL loads a table of the ASCII characters 128 through 255 into memory. If you have a color or graphics adapter, this table lets you display foreign language characters when you are in graphics mode.

After GRAFTABL loads the character table, it displays the message:

GRAPHICS CHARACTERS LOADED

You can load the graphics table only once each time you start MS-DOS. If you try to load the graphics table a second time, GRAFTABL displays the following message:

GRAPHICS CHARACTERS ALREADY LOADED

**NOTE**

This command increases the size of MS-DOS resident in memory.

## GRAPHICS

---

**Purpose:** Lets you print a graphics display screen on a printer when you are using a color or graphics monitor adapter.

**Format:** GRAPHICS [*printer*] [/R] [/B]

where *printer* specifies the type of printer. Printer may be one of the following:

*COLOR1* prints on an IBM Personal Computer Color Printer with black ribbon.

*COLOR4* prints on an IBM Personal Computer Color Printer with RGB (red, green, blue, and black) ribbon.

*COLOR8* prints on an IBM Personal Computer Color Printer with CMY (cyan, magenta, yellow, and black) ribbon.

*COMPACT* prints on an IBM Personal Computer Compact Printer.

*GRAPHICS* prints on an IBM Personal Graphics Printer. *GRAPHICS* is the default.

**Switches:**

- /R** prints black and white (as seen on the monitor) on the printer. The default is to print black as white and white as black.
- /B** prints the background in color. This option is valid for COLOR4 and COLOR8 printers.

**Remarks:**

To print the screen, press < Shift > and < Print Screen > at the same time. If the computer is in 320 x 200 color graphics mode, and if the printer type is COLOR1 or GRAPHICS, GRAPHICS prints the screen contents with up to four shades of gray. If the computer is in 640 x 200 color graphics mode, GRAPHICS prints the screen contents on the paper sideways.

**NOTE**

This command increases the size of MS-DOS resident in memory.

## JOIN

---

**Purpose:** Joins a disk drive to a specific path.

**Type:** External

**Format:** JOIN d1: d2:path [/D]

where:

*d1:* is the source drive .

*d2:path* is the joined path.

**Switch:**

/D turns off the JOIN command. You can specify this switch immediately after the drive name.

**Remarks:**

If the path does not exist, MS-DOS tries to make a directory with that path. The path must be empty. After you issue the JOIN command, the first drive name becomes invalid, and if you try to use it, MS-DOS displays the error message, "Invalid drive."

The JOIN command removes the distinction that physical drives are separately addressable by drive letter. This way you can always refer to all the directories on a specific drive by one path name. If the path already existed before the JOIN command was typed, it will not be usable while the join is in effect.

You can join a drive only at the root. The following command will work:

```
JOIN D: C:\MEMOS
```

The following command will NOT work:

```
JOIN D: C:\MEMOS\JUNE
```

To deassign a splce ("unjoin"), use the following format:

```
JOIN D: /D
```

If you just type:

```
JOIN
```

MS-DOS displays the current drives that are joined.

### **Messages:**

Directory not empty

You can only join onto a directory that is empty.

Incorrect number of parameters

You specified too many or too few options in the command line.

Not enough memory

There is not enough memory for MS-DOS to run the command.

## KEYbxx

---

**Purpose:** Replaces resident ROM BIOS keyboard program with national keyboard programs.

**Type:** External

**Format:** KEYBxx [/N]

where xx specifies one of the following DOS keyboard programs:

*BE* for Belgium  
*DK* for Denmark  
*FR* for France  
*GR* for Germany  
*IT* for Italy  
*NO* for Norway  
*PO* for Portugal  
*SF* for Swiss French  
*SG* for Swiss German  
*SP* for Spain  
*SU* for Finland  
*SV* for Sweden  
*UK* for United Kingdom

**Switches:**

/N causes numeric lock on an enhanced keyboard to be turned off when the program is run.



### Remarks:

After loading a national keyboard program, you can change to the U.S. keyboard by pressing the following function keys:

< Control > - < Alt > - < F1 > to change to the U.S. keyboard program.

< Control > - < Alt > - < F2 > to return to the national keyboard program.

Load only one keyboard program after you start DOS. The program remains resident in memory until you reboot or turn off your computer. If you load a second keyboard program, the new program will define the keystrokes you type. However, the first keyboard program remains in memory but cannot be recalled.

National keyboards contain the following special characters:

*Programmer's Characters.* To use, simultaneously press < Altgr > and the appropriate character key.

*Diacritic Characters.* To use, press the diacritical mark key. Then press the letter key. To use only the diacritical mark, press the key and then the spacebar.

If you use KEYBxx with a computer that has multilingual keyboard support in ROM BIOS, the utility sets the ROM keyboard to the appropriate keyboard and then exits leaving no program resident.

Keycap kits for international applications are available.

## LABEL

---

**Purpose:** Creates changes, or deletes the volume identification label on a disk.

**Type:** External

**Format:** LABEL [D:] [VOLUME LABEL]

where:

*d:* specifies the working drive.

*volume label* specifies the label you want to use to identify the disk.

**Remarks:**

The volume label can be up to 11 characters long. The LABEL command truncates the volume label if you enter more than 11 characters.

You should not use the following characters in a volume label:

\* ? / \ | . , ; : + = < > [ ]

The volume label may include spaces, but may not include tabs.

If you do not specify a volume label, LABEL prompts:

Volume in drive X is xxxxxxxxxx Volume label (11 characters, ENTER for none)?

If the disk does not already have a volume label, LABEL prompts:

Volume in drive X has no label Volume label (11  
characters, ENTER for none)?

Type the volume label that you want and press <Enter>.

If you want to delete the volume label, just press <Enter>.

LABEL prompts with the message:

Delete current volume label (Y/N)?

If you press Y, LABEL deletes the volume label on the disk. If you  
press N, the volume label remains unchanged.

### Messages:

Invalid characters in volume label Volume label (11 characters,  
ENTER for none)?

You cannot use the \* ? / \ | . , ; : + = < > [ ]  
characters or the tab character in a volume label. Reenter  
the volume label.

Cannot LABEL a SUBSTed or ASSIGNed drive

You cannot label a drive that has been assigned via the  
ASSIGN command or that has been substituted with a  
string via the SUBST command.

Cannot label a network drive

You cannot label a drive that is shared on a network  
server station.

Invalid drive specification

You cannot label a drive that is not logically assigned.

## MKDIR (Make directory)

---

**Purpose:** Makes a new directory.

**Type:** Internal

**Format:** MKDIR [d:]path

or

MD [d:]path

where *d:path* specifies the drive and name of the new directory.

### Remarks:

This command creates a multilevel directory structure. When you are in your root directory, you can create subdirectories by using the MKDIR command. The command:

```
MKDIR \USER
```

creates a subdirectory \USER in your root directory. To create a directory named JOE under \USER, type:

```
MKDIR \USER\JOE
```

When you create directories with MKDIR, they always appear under your working directory unless you explicitly specify a different path name with MKDIR.

## Messages:

### Unable to create directory

MS-DOS could not create the directory you specified.  
Check to see that there is not a name conflict. You may  
have a file by the same name, or the disk may be full.

## MODE

---

**Purpose:** Sets operation modes for devices.

**Type:** External

**Format:** *For setting printer options:*

MODE LPT#[:][n][, [m][, P]]

where:

# is the printer number. Valid values are 1, 2, and 3.

*n* is the characters per line. Valid values are 80 and 132.

*m* is the lines per inch. Valid values are 6 and 8.

*P* is a switch specifying continuous retry on time-out errors.

*For switching display adapters:*

MODE *n*

or

MODE [*n*],*m*[,*T*]

where:

*n* specifies the drive, characters per line, and monitor adapter. Valid values are as follows:

*40* sets 40 characters per line for a color/graphics monitor adapter.

*80* sets 80 characters per line for a color/graphics monitor adapter.

*BW40* switches the active display adapter to the color/graphics monitor, and sets the display mode to black and white with 40 characters per line.

*BW80* switches the active display adapter to the color/graphics monitor adapter and sets the display mode to black and white with 80 characters per line.

*CO40* switches the active display adapter to the color/graphics monitor, enables color, and sets the display width to 40 characters per line.

*CO80* switches the active display adapter to the color/graphics monitor, enables color, and sets the display width to 80 characters per line.

*MONO* switches the active display adapter to the monochrome display adapter, which has a display width of 80 characters per line.

*m* specifies shift display.

*T* request a test pattern to align the display.

*For asynchronous communications adapter:*

```
MODE
COMn[:]baud, [parity], [databits],
[stopbits, P]]]
```

where:

*n* is the adapter number. Valid values are 1 and 2.

*baud rate* is 110, 150, 300, 600, 1200, 2400, 4800, or 9600, whichever is appropriate.

*parity* is N (none), O (odd), E (even), whichever is appropriate. The default is E.

*data bits* is either 7 or 8. The default is 7.

*stop bits* is either 1 or 2. The default value is 1.

*P* indicates an asynchronous adapter is being used for a serial interface printer. If you specify *P*, MODE continuously retries time-out errors. To stop the retry loop press <Ctrl> - <Break>.



*Redirecting parallel printer output to an asynchronous communications adapter:*

MODE LPT#[:]=COMn

where:

# specifies the printer; use 1, 2, or 3.

n specifies asynchronous communication adapter number; use 1 or 2.

### Comments:

Before you can use MODE to redirect parallel printer output to a serial device, you must set up the asynchronous communications adapter.

MODE makes printer and asynchronous adapter code resident in memory, increasing the amount to DOS in memory.

The following command disables the redirection for the printer designated by the #.

MODE LPT#[:] [n] [,m]

**Examples:**

To set the mode of operation of printer #1 to 132 characters per line and 8 lines per inch, type:

```
MODE LPT1:132,8
```

To set the mode of operation to 80 characters per line and shift the display two characters to the right, type:

```
MODE 80,R,T
```

MODE will display the test pattern again, without reentering the command. Since you specified T in the MODE command, a prompt ask you if the screen is aligned properly. Enter Y if the alignment is correct. MODE processing is terminated. Enter any character other than Y to repeat the shift.

To set the mode of operation for adapter COM1 to a 1200 baud rate, no parity, 8 databits, and 1 stopbit, type:

```
MODE COM1:12,N,8,1,P
```

To set the mode of operation for adapter COM 1 to the default values (even parity, 7 data bits, 1 stopbit), type:

```
MODE COM1:12,,,P
```

## MORE

---

**Purpose:** Sends output to the console one screen at a time.

**Type:** External

**Format:** MORE

**Remarks:**

MORE is a filter that reads from standard input (such as a command from your terminal) and displays one screen of information at a time. The MORE command then pauses and displays the --More-- message at the bottom of your screen.

Pressing any key displays another screen of information. This process continues until all the data has been read.

The MORE command is useful for viewing a long file one screen at a time. If you type:

```
TYPE MYFILES.NEW | MORE
```

MS-DOS displays the file MYFILES.NEW one screen at a time.

## PATH

---

**Purpose:** Sets a command search path.

**Type:** Internal

**Format:** PATH [[d:][path]; [d:][path2]...]

where:

*d:path1* specifies primary path searched.

*d:path2 . . .* specifies additional paths to be searched.

**Remarks:**

This command tells MS-DOS which directories should be searched for external commands with extensions of .BAT, COM, or EXE after your working directory has been searched. The default value is no path.

To search the \USER\JOE directory for external commands, type:

```
PATH \USER\JOE
```

MS-DOS now searches the \USER\JOE directory for external commands until you set another path or exit MS-DOS.

You can tell MS-DOS to search more than one path by specifying several paths separated by semicolons. For example:

```
PATH C:\USER\JOE;B:\USER\SUE;\BIN\DEV
```

tells MS-DOS to search the directories specified by the above path to find external commands. The paths are searched in the order specified in the PATH command.

The command, PATH, with no options prints the current path.

If you specify:

```
PATH ;
```

MS-DOS clears all previously-defined paths. This means that only the working directory is searched for external commands.

### **Messages:**

#### **No path**

You typed PATH with no options but have not set a command search path.

## PRINT

---

**Purpose:** Prints text files on a line printer while you are processing other commands (usually called "background printing").

**Type:** External

**Format:** PRINT [D:][path][filename[.ext]]  
[/D:device] [/B:size] [/Q:value]  
[/T] [/C] [/P]

where *d:path filename.ext* specifies file to be printed.

**Switches:**

The following switches are provided with this command:

/D specifies the print device. If not specified, the default device is PRN. PRINT prompts for a print device.

The following switches are allowed only the first time you run the print command after starting MS-DOS.

/B sets the size in bytes of the internal buffer. Increasing the value of /B speeds up the PRINT command.

/Q specifies the number of files allowed in the print queue if you want more than 10. The minimum value for the /Q switch is 4; the maximum is 32.

- /T** deletes all files in the print queue (those waiting to be printed). A message to this effect will be printed.
- /C** turns on cancel mode. The preceding file name and all following file names are removed from the print queue.
- /P** turns on print mode. The preceding file name and all following file names are then added to the print queue.

**Remarks:**

Use the PRINT command only if you have a line printer attached to your computer.

If you use PRINT without options, it displays the contents of the print queue on your screen without affecting the queue.

**Examples:**

```
PRINT /T
```

Empties the print queue.

```
PRINT A:TEMP1.TST /C A:TEMP2.TST  
A:TEMP3.TST
```

Removes the three files indicated from the print queue.

```
PRINT TEMP1.TST /C TEMP2.TST /P TEMP3.TST
```

Removes TEMP1.TST from the queue, and adds TEMP2.TST and TEMP3.TST to the queue.

## NOTE

Each print queue entry may contain a maximum of 64 characters, including the drive name. Therefore, you may need to change directories if you want to print files in deep subdirectories.

### Messages:

#### All files canceled

If you specify the /T ("terminate") switch, MS-DOS prints "All files canceled by operator" on your printer. If the current file being printed is canceled by /C, the message "File canceled by operator" is printed.

#### Cannot open (filename)

Either MS-DOS cannot find the specified file to print or the file does not exist. Check the command for a valid file name.

Errors on list device indicate that it may be offline. Please check it  
Your printer is turned off.

#### (filename) file not found

You switched disks when a file was queued up, but before it started to print. Reissue the PRINT command for that file.

#### List output is not assigned to a device

This message is displayed if the "Name of list device" specified to the above prompt is invalid. Subsequent attempts return the same message until you specify a valid device.



Name of list device [PRN:]

This prompt appears when PRINT is run the first time and the \D switch is not specified. Any current device may be specified and that device then becomes the PRINT output device. As shown in the brackets, if you press <Enter> only, MS-DOS uses the default list device (PRN).

No files match drive:xxxxxxxx.xxx

A drive and file name were given for files to add to the queue, but no files match.

**NOTE**

If there are no files in the queue to match the canceled file name, no error message appears.

Print queue is empty

There are no files in the print queue.

Print queue is full

There is room for 10 files in the queue. If you attempt to put more than 10 files in the queue, this message appears on the screen. To add more files to the queue, refer to the /Q switch in the list above.

Resident part of Print installed

This is the first message that MS-DOS displays when you issue the PRINT command. It means that available memory has been reduced by several thousand bytes to process the PRINT command along with other processes.

## PROMPT

---

**Purpose:** Changes the MS-DOS command prompt.

**Type:** Internal

**Format:** PROMPT [text]

where *text* represents the new command prompt.

**Remarks:**

This command allows you to change the MS-DOS system prompt (for example, A > ). If no text is typed, the prompt is set to the default prompt, which is the default drive designation. You can set the prompt to a special prompt, such as the current time, by using the characters indicated below.

You can use the following characters in the prompt command to specify special prompts. They must all be preceded by a dollar sign (\$):

Specify This Character	To Get This Prompt
\$	The '\$' character
T	The current time
D	The current date
P	The working directory of the default drive
V	The version number
N	The default drive
G	The '>' character
L	The '<' character
B	The ' ' character
-	A Return-line feed sequence
S	A space (leading only)
E	ASCII code X'1B' (escape)

**Examples:**

```
PROMPT $P
```

Sets the drive prompt to drive: current directory.

```
PROMPT TIME = %T%_DATE = %D
```

Sets a two-line prompt which prints: Time = (current time)  
Date = (current date)

If your terminal has an ANSI escape sequence driver, you can use escape sequences in your prompts. For example:

```
Prompt %e[7m%n:%e[m
```

Sets prompts in reverse video mode and returns to video mode for other text.

## RECOVER

---

**Purpose:** Recovers a file or an entire disk containing bad sectors.

**Type:** External

**Format:** RECOVER [d:]

or

RECOVER d:[path][filename[.ext]]

where *d:path filename.ext* is the drive, path, and file to be recovered.

**Remarks:**

If a sector on a disk is bad, you can recover either the file containing that sector (without the bad sector) or the entire disk (if the bad sector was in the directory).

To recover a particular file, type:

RECOVER filename

This causes MS-DOS to read the file sector by sector and to skip the bad sector(s). When MS-DOS finds the bad sector(s), the sector(s) are marked and MS-DOS will no longer allocate your data to that sector.

To recover a disk, type

RECOVER d:

where *d:* is the letter of the drive containing the disk to be recovered.

**Messages:**

File not found

MS-DOS cannot find the file that you specified. Check to see that the path name is accurate and that the file exists in the directory you specified.

(xxxx) of (xxxx) bytes recovered

This message tells you the number of bytes that MS-DOS was able to recover from the disk.

Warning directory full

The root directory is too full for RECOVER processing. Delete some files in the root directory to free space.

## REN (Rename)

---

**Purpose:** Changes the name of a file.

**Type:** Internal

**Format:** REN [d:][path] filename1.[ext]  
filename2.[ext]

where:

*d:path filename1.ext* is the old file name. You must specify *d*: If the file does not reside on the default drive.

*filename2.ext* is the new file name.

### Remarks:

You cannot rename files across drives or directories. If a drive or path is entered for filename2.ext, it is ignored.

Wildcards may be used in either option. All files matching the first file name are renamed. If wildcards appear in the second file name, corresponding character positions will not be changed.

For example, the following command changes the extension of all file names ending in .LST to .PRN:

```
REN *.LST *.PRN
```

**Example:**

To rename the file CHAP10 on drive B to PART10:

```
REN B:CHAP10 PART10
```

The file remains on drive B.

**Messages:**

Duplicate file name or file not found

You tried to rename a file to a name already present in the directory or the file you are trying to rename could not be found.



## REPLACE

---

**Purpose:** Add or replace files on a target with files from a source.

**Type:** External

**Format:** REPLACE[d1:][path]filename[.ext]  
[d2:][path][ /A ][ /P ][ /R ][ /S ][ /W ]

where:

*d1: path filename.ext* specifies files on the source that are to be added to or replaced on the target. Global characters can be used in the filename.

*d2: path* is the drive and path of the target.

The target drive, path, and filename string can not exceed 64 characters.

**Switches:**

/A      copies specified files from the source that do not exist on the target. /A cannot be used with /S.

/P      prompts before each specified file found on the target is added or replaced.

- /R** replaces all specified files on the target.
- /S** searches all the directories on the target for the files that match the source file. **/S** cannot be used with **/A**.
- /W** waits for the user to insert a diskette before searching for source files. If **/W** is not used the search begins immediately.

**Example:**

Assume you changed your file README.DOC in your subdirectory C:\UPDATES. To replace all copies of this file on your fixed disk, type the following command:

```
REPLACE C:\UPDATES\README.DOC C:\ /S
```

## RESTORE

---

**Purpose:** Restores BACKUP files from either floppy disk to hard disk or hard disk to floppy disk.

**Type:** External

**Format:** RESTORE d1: [d2:][path]  
[filename[.ext]][/S[/P]]

where:

*d1:* is the drive containing the BACKUP files.

*d2:path filename.ext* are the files you want to restore. If you do not specify a path, the utility restores files belonging to the current directory.

Global filename characters cause RESTORE to process all files matching the filename in the current directory or specified path.

### Switches:

*/S* restores backup files in all subdirectories below the current directory level in addition to the specified directory files.

*/P* prompts you to choose whether or not to restore a read-only or modified file. By using */P* you can prevent RESTORE from writing over a file you have modified after you backed it up.

**Comments:**

RESTORE sets the following:

Error Level Explanation

- 0 Normal completion
- 1 No files found to process
- 2 Unable to process some files due to sharing conflicts
- 3 User pressed <Ctrl>-<Break> or <Esc> to abort
- 4 Error caused termination

**Examples:**

The following command restores all files on the backup floppy disks to fixed disk drive C:.

```
RESTORE A: C:\*.* /S
```

The following commands restore three different files from backup floppy disks to the default hard disk drive.

```
RESTORE A: \LEVEL1\MYFILE1.DAT
```

```
RESTORE A: \LEVEL1\LEVEL2\MYFILE2.DAT
```

```
RESTORE A: \LEVEL1\LEVEL3\MYFILE3.DAT
```

The following command restores all .TXT files from the floppy disk in drive A: to the hard disk.

```
RESTORE A: C:*.TXT
```

## RMDIR (Remove directory)

---

**Purpose:** Removes a directory from a multilevel directory structure.

**Type** Internal

**Format:** RMDIR path

or

RD path

where *path* is the directory to delete.

**Remarks:**

This command removes a directory that is empty except for the . and .. shorthand symbols. You must delete all files in the directory first.

**Example:**

To remove the \USER\JOE directory, first issue a DIR command for that path to ensure that the directory is empty. Then type:

```
RMDIR \USER\JOE
```

The directory will be deleted from the directory structure.

## SELECT

---

**Purpose:** Creates DOS on a target disk with the keyboard layout and the date and time format of your choice.

**Type:** External

**Caution:** Since SELECT uses the FORMAT command, all data on the target disk is destroyed. Before any data is erased, SELECT asks the user if he or she wants to continue.

**Format:** SELECT [[A:|B:] D:[path]] xxx yy

where:

*A:|B:* specifies the source drive and path. If source drive is specified, a target drive must be entered. The default source drive is A:.

*D:path* specifies the target drive and path. The target and source drives cannot be the same. The default target drive is B. The default path is the target drive's root directory.

*xxx* is the country code. The country code sets the date and time format, the currency symbol, and the decimal separator.

*yy* specifies the keyboard code. The keyboard code sets the keyboard layout. The keyboard code must match a corresponding KEYBxx.COM file.

All paths must be specified from the root directory. Do not use the . or .. entries to specify paths.

**Remarks:**

SELECT uses XCOPY to make a copy of your DOS diskette in drive A: and creates the following two files:

CONFIG.SYS file that contains COUNTRY = command.

AUTOEXEC.BAT file that contains the KEYBxx command, files created only for France, Germany, Italy, Spain, and United Kingdom.

SELECT boots the new keyboard through the AUTOEXEC.BAT file.

The following table lists the country codes and keyboard codes for countries that SELECT accepts:

COUNTRY	COUNTRY CODE	KEYBOARD CODE
United States	001	US
France	033	FR
Spain	034	FR
Italy	039	IT
United Kingdom	044	UK
Germany	049	GR

No code is necessary for the United States because it is the default keyboard layout. If a country is not listed, manually edit AUTOEXEC.BAT to include the keyboard code.

## SET

---

**Purpose:** Sets one string value in the environment equal to another string for use in later programs.

**Type:** Internal

**Format:** SET [string1=string2]

where:

*string1* specifies current string value.

*string2* specifies future string value.

**Remarks:**

This command is useful only if you want to set values that will be used by programs you have written.

When MS-DOS sees a SET command, it inserts the entire string into a part of memory reserved for environment strings. If the name exists in the environment, it is replaced by the new string. If you type the SET command with only the first string, the associated string name is removed from the environment. If you type SET with no options, MS-DOS displays the current environment settings.

An application program can get a listing of all environment values that have been set by examining its environment. (Environments are passed in the Program Segment Prefix).



The SET command can also be used in batch processing. In this way, you can define your replaceable parameters with names instead of numbers. If your batch file contains the statement "LINK %FILE%", you can set the name that MS-DOS will use for that variable with the SET command. The command

```
SET FILE = DOMORE
```

replaces the %FILE% parameter with the file name DOMORE. Therefore, you do not need to edit each batch file to change the replaceable parameter names. Note that when you use text (Instead of numbers) as replaceable parameters, the name must be ended by a percent sign.

### Examples:

The command

```
SET TTY=VT52
```

sets your TTY value to VT52 until you change it with another SET command. If you just type:

```
SET
```

MS-DOS displays the current environment strings.

## SHARE

---

**Purpose:** Installs file sharing and locking.

**Type:** External

**Format:** SHARE [/F:space] [/L:locks]

**Switches:**

*/F:space*

allocates file space (in bytes) for the area MS-DOS uses to record filesharing information. Each file that is open needs the length of the full file name plus 11 bytes (the average name is 20 bytes). The default value for the /F switch is 2048 bytes.

*/L:locks*

allocates the number of locks you want to allow. The default value for the /L switch is 20 locks.

**Remarks:**

The SHARE command is only used when networking is active. It is included in the AUTOEXEC.BAT file to install shared files. Refer to the *AST Network Manager's Guide* to learn about shared files.

Once you have used the SHARE command in an MS-DOS session, all read and write requests are checked by MS-DOS.

**Example:**

The following example loads file sharing and uses the default values for the /F and /L switches:

SHARE

**Messages:**

Incorrect parameter

One of the options you specified is wrong.

Not enough memory

There is not enough memory for MS-DOS to run the command.

Share Already Installed

You can install Share only once.

## **Sort**

---

**Purpose:** SORT reads standard input, sorts the data, then writes it to your terminal screen, a file, or to another device.

**Type:** External

**Format:** SORT [d:] [path] [filename [.ext]]  
[/R] [/+N]

where *d:path filename.ext* is the file to be sorted.

**Switches:**

/R      reverse the sort; that is, sort from Z to A.

/+N     SORT starting with column n where n is some number. If you do not specify this switch, SORT will begin sorting from column 1.

**Remarks:**

Sort can be used, for example, to alphabetize a file by a certain column. There are two switches which allow you to select options:

**Examples:**

This command reads the file UNSORT.TXT, reverses the sort, and then writes the output to a file named SORT.TXT:

```
SORT /R UNSORT.TXT SORT.TXT
```

The following command pipes the output of the directory command to the SORT filter. The SORT filter sorts the directory listing starting with column 14 (this is the column in the directory listing that contains the file size), then sends the output to the screen. The result of this command is a directory sorted by file size:

```
DIR | SORT /+14
```

The command:

```
DIR | SORT /+14 | MORE
```

does the same thing as the command in the previous example, except that the MORE filter gives you a chance to read the sorted directory one screen at a time.

## SUBST

---

**Purpose:** Substitutes a virtual drive for a path name.

**Type:** External

**Format:** SUBST [d:] [path] [/D]

where:

*d:* is the virtual drive.

*path* is directory/subdirectory being substituted.

**Switch:**

/D deletes an associated drive or path. You can specify this switch immediately after the drive name.

**Remarks:**

Use when an application program will not accept a pathname but will accept a drive letter.

Allowance for the virtual drive must be made in your CONFIG.SYS file before using SUBST. (See LASTDRIVE command in CONFIG.SYS.)

Path must be to an existing directory/subdirectory. SUBST will not create a directory/subdirectory.

When MS-DOS executes a command, it replaces the virtual drive entered in the command with the path entered in the SUBST command.

If you type:

**SUBST**

MS-DOS displays the names of the substitutions in effect.

Substitution remains effective until the system is booted or the /D switch is used.

**Example:**

The command:

**SUBST Z: B:\USR\FRED\FORMS**

substitutes virtual drive Z: for the pathname B:\USR\FRED\FORMS. Now, instead of typing the full path, you can get to this directory by simply typing Z:.

**Messages:**

Incorrect number of parameters

You specified too many or too few options in the command.

Not enough memory

There is not enough memory for MS-DOS to run the command.

## SYS (System)

---

**Purpose:** Transfers the hidden, MS-DOS system files from the disk in the default drive to the disk in the specified drive.

**Type:** External

**Format:** SYS d:

where *d*: is the drive the system files are transferred to.

**Remarks:**

SYS does not copy the COMMAND.COM file. Use the COPY command to transfer COMMAND.COM to your disk.

SYS is normally used to update the system or to place the system on a formatted disk that contains no files. You must type a drive letter with this command.

If IO.SYS and MSDOS.SYS are on the target disk, they must take up the same amount of space on the disk as the new system will need. This means that you cannot transfer system files from an MS-DOS 2.0 disk to an MS-DOS 1.1 disk. You must reformat the MS-DOS 1.1 disk with the FORMAT command before the SYS command will work. The target disk must be completely blank or have the system files IO.SYS and MSDOS.SYS. The transferred files are copied in the following order:

IO.SYS MSDOS.SYS

IO.SYS and MSDOS.SYS are both hidden files that do not appear when you type the DIR command.



**Messages:**

No room for system on destination disk

There is not enough room on the target disk for the  
IO.SYS and MSDOS.SYS files.

Incompatible system size

The system files IO.SYS and MSDOS.SYS do not take up  
the same amount of space on the target disk as the new  
system will need.

## TIME

---

**Purpose:** Displays and sets the time.

**Type:** Internal

**Format:** TIME [hours:minutes]

where:

*hours* is current hour.

*minutes* is current minute.

**Remarks:**

If you type TIME, the following message is displayed:

Current time is (hh):(mm):(ss).(cc) Enter new time:

Press <Enter> If you do not want to change the time shown. If you want to change the time after you have started MS-DOS, (for example, to 8:20 a.m.), type:

TIME 8:20

In response to the MS-DOS prompt. The new time must be entered using numbers only; letters are not allowed. The allowed options are:

HOURS = 00-24 MINUTES = 00-59

Separate the hour and minute entries by a colon. You do not have to type the ss (seconds) or cc (hundredths of seconds).

If you do not type a valid time, MS-DOS displays the message:

Invalid time Enter new time:

MS-DOS then waits for you to type a valid time.

Like the DATE command, the TIME command format can be changed with the COUNTRY command in the CONFIG.SYS file.

## TREE

---

**Purpose:** Displays the path (and optionally list the contents) of each directory and subdirectory on the given drive.

**Type:** External

**Format:** TREE [d:] [/F]

where *d*: specifies the drive you want to use. If you do not specify this option, TREE uses the default drive.

**Switch:**

/F displays the names of the files in each directory.

**Remarks:**

The TREE command lists the full path of each directory, along with the names of each of their subdirectories.

**Example:**

If you want to print a list of the directory and file names on the disk in drive B:, you can use the command:

```
TREE B: /F > PRN
```

**Messages:**

No subdirectories exist

The disk in the drive you specified does not contain subdirectories.

Invalid path

You specified an invalid path.

Invalid drive specification

The drive name that you specified does not exist or is invalid.

Invalid parameter

The /F switch is the only valid parameter for TREE.

Incorrect DOS version

You are using the TREE command with the wrong version of MS-DOS. Insert the correct version of DOS and try the command again.

## TYPE

---

**Purpose:** Displays the contents of a file on the screen.

**Type:** Internal

**Format:** TYPE [d:][path]filename[.ext]

where *d:path filename.ext* specifies file to be displayed.

**Remarks:**

Use this command to view a file without modifying it. (Use DIR to find the name of a file and EDLIN to change the contents of a file.)

When you use TYPE to display a file with tabs in it, all tabs are expanded to 8 spaces wide. A display of binary files causes control characters (such as Control-Z) to be sent to your computer, including bells, form feeds, and escape sequences.

## VER (Version)

---

**Purpose:** Prints MS-DOS version number.

**Type:** Internal

**Format:** VER

**Remarks:**

If you want to know what version of MS-DOS you are using, type:

VER

The version number will be displayed on your screen.

## VERIFY

---

**Purpose:** Turns the verify switch on or off when writing to disk.

**Type:** Internal

**Format:** VERIFY [ON]  
  
or  
  
VERIFY [OFF]

**Remarks:**

This command has the same purpose as the V switch in the COPY command. If you want to verify that all files are written correctly to disk, you can use the VERIFY to make sure your files are intact (no bad sectors, for example). MS-DOS will perform VERIFY each time you write data to a disk. You will receive an error message only if MS-DOS was unable to successfully write your data to disk.

VERIFY ON remains in effect until you change it in a program (by a SET VERIFY system call), or until you type VERIFY OFF.

If you want to know the current VERIFY setting, type:

VERIFY



## VOL (Volume)

---

**Purpose:** Displays disk volume label, if it exists.

**Type:** Internal

**Format:** VOL [d:]

where *d*: specifies drive of volume label to be displayed.

**Remarks:**

This command prints the volume label of the disk in a specific drive. If you do not type a drive letter, MS-DOS prints the volume label of the disk in the default drive.

**Messages:**

Volume in drive x has no label  
The disk in the drive does not have a volume label.

## XCOPY

---

**Purpose:** Copies files and directories, including lower level directories if they exist.

**Type:** External

**Format:** XCOPY [d1:][path]filename[.ext]  
[d2:][path][filename[.ext]] [/A]  
[/D] [/E] [/M] [/P] [/S] [/V] [/W]

or

XCOPY [d1:] path [filename[.ext]]  
[d2:][path][filename[.ext]] [/A]  
[/D] [/E] [/M] [/P] [/S] [/V] [/W]

or

XCOPYd1: [path][filename[.ext]]  
[d2:][path][filename[.ext]] [/A]  
[/D] [/E] [/M] [/P] [/S] [/V] [/W]

where:

*d1: path filename.ext* is the start point of the files to be copied. The start point can be any combination of the drive, file, or directory parameters.

*d2: path filename.ext* is the target drive, path, or filename for the files being copied.

You cannot use the reserved devices CON and LTP1 as filenames.

You cannot use more than 64 characters to define the source or target drive, path, and filename.

The default start directory is the current directory. the default file name is \*.\*.

Switches:

- /A      copies files that have their archive bit set. It does not modify the archive bit of the source file. A set archive bit set Indicates the file has been created or changed since the last XCOPY /M.
- /D      copies source files you modified on or after the date specified by <mm-dd-yy>. The date format must match the date format selected in either the SELECT or COUNTRY commands. For example, /D:05/15/87 copies files later than May 15, 1987.
- /E      creates empty subdirectories on the target that may result from the copy procedures. If /E is not used, empty subdirectories are not created. /E works only in conjunction with /S.
- /M      copies files whose archive bit is set. A set archive bit indicates the file has been created or changed since the last XCOPY /M. When the copy procedure is complete, /M turns the the archive bit off.

- /P** prompts you with "(Y/N?)" before copying each file.
- /S** copies all files in and below the source directory. If /S is not used, only files within the source directory are copied.
- /V** verifies all sectors written to the target diskette are correct.
- /W** waits for you to insert a diskette before searching for source files. If /W is not used, the search begins immediately.

**Remarks:**

Be sure the target space is large enough to hold the copied files. If the target space is too small, XCOPY displays a disk full message and the program ends. To perform the XCOPY, insert another diskette and reissue the XCOPY command.

Unlike files copied with the BACKUP command, files on the target diskette copied with the XCOPY command can be accessed easily.

### Example:

To copy all files that have been changed on drive C: to a formatted diskette in drive A:, type:

```
XCOPY C:\*.* A: /S /M
```

In the above example, all files on C: with an archive bit equal to 1 will be copied to A:. After a file is copied, the file's archive bit is set to 0 on drive C: (the source drive.)

To copy all files with an extension of .DAT and created or modified on or after April 15, 1987 from A:\JOHN to C:\JEFF, type:

```
XCOPY A:\JOHN\*.DAT C:\JEFF /D=04/15/87
```

### Messages:

Cannot perform a cyclic copy

When you are using the /S switch, you may not specify a target that is a subdirectory of the source.

### 3.4 Batch Processing Commands

The following commands are called batch processing commands. They can add flexibility and power to your batch programs.

The commands discussed are:

ECHO

FOR

GOTO

IF

PAUSE

SHIFT

## ECHO

---

**Purpose:** Turns batch echo feature on and off.

**Format:** ECHO [ON]  
  
or  
  
ECHO [OFF]  
  
or  
  
ECHO [message]

**Remarks:**

Normally, commands in a batch file are displayed (echoed) on the screen when they are seen by MS-DOS. The command:

ECHO OFF

turns off this feature. ECHO ON turns the echo back on.

If neither ON nor OFF is specified, the current setting is displayed.

ECHO message is only useful if ECHO is off and you are using a batch file. By typing ECHO and a message in your batch file, you can print messages on the screen.

## FOR

---

**Purpose:** Command extension used in batch and interactive file processing.

**Format:** *For batch processing:*

FOR %% c in set do command

*For interactive processing:*

FOR % c in set do command

where:

c can be any character except 0,1,2,3,...,9 (to avoid confusion with the %0-%9 batch parameters).

set is a list of files to be processed

**Remarks:**

The %%c variable is set sequentially to each member of *set*, and then *command* is evaluated. If a member of *set* is an expression involving \* and/or ?, then the variable is set to each matching pattern from disk. In this case, only one such item may be in the set, and any item besides the first is ignored.



**Examples:**

```
FOR %%F IN ( *.ASM ) DO MASM %%F;
```

```
FOR %%F IN (REPORT MEMO ADDRESS) DO DEL  
%%F
```

The first example binds the variable named %F to files ending with \*.ASM in the working directory. It then executes the command:

```
MAASM FILENAME
```

where *filename* could be BIGFILE.ASM SORTER.ASM LIST.ASM

The second example binds the variable %F to the files named report, memo, and address. It then deletes each of these files.

The %% is needed so that after batch parameter (%0-%9) processing is done, there is one % left. If only %F were there, the batch parameter processor would see the %, look at f, decide that %F was an error (bad parameter reference) and throw out the %F, so that the command FOR would never see it. If the FOR is not in a batch file, then only one % should be used.

**Messages:**

For cannot be nested

Nesting of FOR statements is illegal.

## GOTO

---

**Purpose:** Command extension used in batch file processing.

**Format:** GOTO label

**Remarks:**

GOTO causes commands to be taken from the batch file beginning with the line after the label definition. If no label has been defined, the current batch file will terminate.

**Example:**

```
:ONE REM FIRST PROGRAM GOTO TWO  
...  
:TWO REM SECOND PROGRAM
```

sends the program processor to the label named TWO.

Starting any line in a batch file with a colon (:) causes the line to be ignored by batch processing. The characters following GOTO define a label.

## IF

---

**Purpose:** Command extension used in batch file processing.

**Format:** IF condition command

where:

*condition* is one of the following:

*ERRORLEVEL number*

True if, and only if, the previous program executed by COMMAND.COM had an exit code of number or higher. (An exit code is set by a specific program. It is returned by the operating system after the program has finished. Later program tasks may be performed based on the value of this number.)

*string1 == string2*

True if, and only if, *string1* and *string2* are identical after parameter substitution. Strings may not have embedded separators.

*EXIST filename*

True if, and only if, *filename* exists.

*NOT condition*

True if, and only if, *condition* is false.

*command* is any internal or external MS-DOS command, such as DIR/W or CHKDSK.

**Remarks:**

The IF statement allows conditional execution of commands. When the *condition* is true, then the *command* is executed. Otherwise, the *command* is ignored.

**Examples:**

```
IF NOT EXIST DATA1 ECHO CAN'T FIND FILE  
IF NOT ERRORLEVEL 3 LINK $1,,;
```

```
IF %1==JOHN GOTO L1  
.  
.  
.  
:L1
```

## PAUSE

---

**Purpose:** Suspends execution of the batch file.

**Format:** PAUSE [comment]

**Remarks:**

When a batch file is running, you may need to change disks or perform some other action. Pause suspends execution until you press any key, except <Ctrl>-<C> or <Ctrl>-<Break>.

When the command processor encounters PAUSE, it prints:

Strike a key when ready . . .

If you press <Ctrl>-<C> or <Ctrl>-<Break>, MS-DOS displays the following message:

Terminate batch job (Y/N)?

If you type Y in response to this prompt, the batch file ends and control returns to the operating system. Therefore, PAUSE can be used to break a batch file into pieces, allowing you to end the batch command file at any intermediate point.

The comment is optional and may be typed on the same line as PAUSE. You may also want to prompt the user of the batch file with some meaningful message when the batch file pauses. For example, you may want to change disks in one of the drives. An optional prompt message may be given in such cases. The comment prompt will be displayed before the "Strike a key" message.

## REM

---

**Purpose:** Displays remarks which are on the same line as the REM command in a batch file during execution of that batch file.

**Format:** REM [comment]

where *comment* represents remark to be displayed.

**Remarks:**

The only separators allowed in the comment are the space, tab, and comma.

**Example:**

```
1: REM THIS FILE CHECKS NEW DISKS
2: REM IT IS NAMED NEWDISK.BAT
3: PAUSE INSERT NEW DISK IN DRIVE B
4: FORMAT B:/S
6: CHKDSK B:
```

You can use REM without a comment in your file to add spacing for readability.

## SHIFT

---

**Purpose:** Allows access to more than 10 replaceable parameters in batch file processing.

**Format:** SHIFT

**Remarks:**

Usually, command files are limited to handling 10 parameters, %0 through %9. To allow access to more than ten parameters, use SHIFT to change the command line parameters. For example if:

```
%0    = TAN
%1    = BAR
%2    = NAME
%3...%9  are empty
```

then a SHIFT results in the following:

```
%0    = BAR
%1    = NAME
%2...%9  are EMPTY
```

If there are more than 10 parameters given on a command line, those that appear after the 10th (%9) will be shifted one at a time into %9 by successive shifts.

### NOTE

There is no backward shift. Once SHIFT is executed, the 0 parameter (%0) that existed before the shift cannot be recovered.







This section explains the MS-DOS editing and function keys.

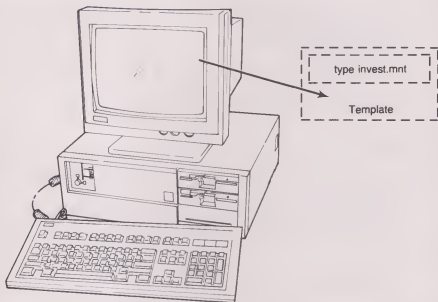
### 4.1 Special Editing Keys

The special editing keys handle command input differently than most operating systems. You do not have to type the same sequences of keys repeatedly, because the last command line is automatically placed in a storage area called a template.

By using both the template and the special editing keys, you can take advantage of the following MS-DOS features:

1. A command line can be instantly repeated by pressing two keys.
2. If you make a mistake in the command line, you can edit and retry without having to retype the entire command line.
3. A command line similar to a preceding command line can be edited and executed with a minimum of typing by pressing a special editing key.

The relationship between the command line and the template is shown in Figure 4-1.



**Figure 4-1. Command Line and Template.**

As seen in Figure 4-1, you type a command to MS-DOS on the command line. When you press <Enter>, the command is sent to the command processor (COMMAND.COM) for execution. At the same time, a copy of the command is sent to the template. The template copy, allows you to recall or modify the command with MS-DOS special editing keys.

**Table 4-1. Special Editing Functions.**

Key	Name	Editing Function
<F1>	<COPY1>	Copies one character from the template to the command line
<F2>	<COPYUP>	Copies characters up to the character specified in the template and puts these characters on the command line.
<F3>	<COPYALL>	Copies all remaining characters in the template to the command line.
<Del>	<SKIP1>	Skips over (does not copy) a character in the template.
<F4>	<SKIPUP>	Skips over (does not copy) the characters in the template up to the character specified.
<Esc>	<VOID>	Voids the current input; leaves the template unchanged.
<Ins>	<INSERT>	Enters/exits insert mode.
<F5>	<NEWLINE>	Makes the new line the new template .
<F6>	<Ctrl>-Z	Puts a <Ctrl>-Z (1AH) end-of-file character in the new template.

### Examples:

If you type:

```
DIR PROG.COM
```

MS-DOS displays information about the file PROG.COM on your screen. DIR PROG.COM is also saved in the template. To repeat the command, just press:

```
< F3 > < Enter >
```

The repeated command is displayed on the screen as you type, as shown below:

```
< F3 > DIR PROG.COM < Enter >
```

Pressing < F3 > copies the contents of the template to the command line; pressing < Enter > sends the command line to the command processor for execution.

If you want to display information about a file named PROG.ASM, use the contents of the template by entering:

```
< F2 > C
```

All characters from the template to the command line, up to but not including C will be copied. MS-DOS displays:

```
DIR PROG.
```

The underline is your cursor. Now type:

ASM

The result is:

DIR PROG.ASM

The command line DIR PROG.ASM is now in the template and ready to be sent to the command processor for execution. To run the command, press <Enter>.

Now, assume you want to run the following command:

TYPE PROG.ASM

To do this, type:

TYPE <Insert> <F3> <Enter>

When you are typing, the characters are entered directly into the command line and overwrite corresponding characters in the template. This automatic replacement feature is turned off when you press the <Insert> key.

Thus, TYPE replaces DIR in the template. To insert a space between TYPE and PROG.ASM, you pressed <Insert> and then the Spacebar.

Finally, to copy the rest of the template to the command line, you pressed <F3> and then <Enter>. The command TYPE PROG.ASM has been processed by MS-DOS, and the template becomes TYPE PROG.ASM.

If you had misspelled TYPE as BYTE, a command error would have occurred. Still, instead of throwing away the whole command, you could save the misspelled line before you press <Enter> by creating a new template with the <F5> key:

```
BYTE PROG.ASM<F5>
```

You could then edit this error by typing:

```
T<F1>P<F3>
```

<F1> copies a single character from the template to the command line. The result is the command you want:

```
TYPE PROG.ASM
```

As an alternative, you can use the same template containing BYTE PROG.ASM and then use <Delete> and <Insert> to get the same result:

```
<Delete> <F1> <Insert> YP <F3>
```

To illustrate how the command line is affected as you type, examine the keys pressed (listed on the left) and their affect (listed on the right):

< Delete >	Skips over 1st template character
< Delete >	Skips over 2nd template character
< F1 > T	Copies 3rd template character
< Insert > YP TYP	Inserts two characters
< F3 > TYPE PROG.ASM	Copies rest of template

< Delete > does not affect the command line. It affects the template by deleting the first character. Similarly, < F4 > deletes characters in the template, up to, but not including, a given character.



## 4.2 Control Character Functions

A control character function is a function that affects the command line. The control character functions are listed in Table 4-2.

**Table 4-2. Control Character Functions.**

Control Character	Function
< Ctrl > - < C >	Aborts current command.
< Ctrl > - < H >	Removes last character from command line, and erases character from terminal screen.
< Ctrl > - < J >	Inserts physical end-of-line, but does not empty command line. Use the Linefeed key to extend the current logical line beyond the physical limits of the terminal screen.

Table 4-2. Continued.

<Ctrl> - <N>	Echoes output to a line printer.
<Ctrl> - <P>	Sends terminal output to a line printer.
<Ctrl> - <S>	Suspends output display on the screen. Press <Ctrl> -S to resume.
<Ctrl> - <X>	Cancels the current line; empties the command line; and then outputs a backslash (\), return, and linefeed. The template used by the special editing commands is not affected.

Remember when you type a control character sequence, such as <Ctrl> - <C>, you must press down <Ctrl> and <C> simultaneously.

## NOTES

EDLIN



This section will show you how to start EDLIN , how to quit EDLIN, and how to use the MS-DOS special editing keys with EDLIN.

## **5.1 About EDLIN**

You can use the MS-DOS line editor, EDLIN, to create text files and save them on you disks, or to update existing files, saving both the original and the updated files. With EDLIN, you can delete, edit, insert, and display lines in files. It will also help you to search for and delete or replace, text within you files.

Though it isn't a word processor, EDLIN does make it easy for you to create and revise files such as memos, letters, reports, or GW-BASIC programs.

### **5.1.1 How EDLIN Works**

EDLIN divides the text from a file into lines, each line containing up to 253 characters. It gives each line a number and always numbers the lines consecutively. Even though you see these line numbers on the screen when you use EDLIN, they are not a part of the program.

When you insert lines of text in a file, the line numbers after the inserted text are automatically adjusted. Similarly, when you delete lines in a file, the line numbers following the deleted text are automatically renumbered.

### 5.1.2 How to Start EDLIN

To start EDLIN, simply type the word EDLIN followed by a file name. If you are creating a new file, the file name should be the name or path name of the file you wish to create. If EDLIN does not find this file on the default disk drive, it creates a new file with the name or path name that you specify. For example, if you want to create a file called BUDGET.MAY, type:

```
EDLIN BUDGET.MAY <Enter>
```

EDLIN displays the following:

```
New File
```

```
*  
—
```

Note the EDLIN prompt is an asterisk (\*).

To begin entering text you must type an I (insert) command to insert lines. The I command is discussed later in this section. For now you can type lines of text into your file, or use any of the EDLIN commands.

#### NOTE

Be sure to press <Enter> at the end of each line.

Suppose you want to edit an existing file called BUDGET.JAN. To do this, you would type:

```
EDLIN BUDGET.JAN <Enter>
```

When EDLIN finds the BUDGET.JAN file, it loads it into memory. If your computer has enough memory to load the entire file, EDLIN displays the following message:

```
End of input file
```

```
*
```

You can then edit the file by using EDLIN commands.

If the file is too large to be loaded into memory, EDLIN loads lines from the file until the memory is three-quarters full and displays the asterisk prompt. You can then edit the portion of the file that is in memory.

To edit the rest of the file, you must save some of the edited lines on a disk to free memory. EDLIN will then be able to load the remaining unedited lines from the disk into memory. Refer to the Write (W) and Append (A) commands.

### 5.1.3 How to Quit EDLIN

When you finish editing, you can save your original and the updated file by using the End (E) command at the \* prompt. To return to the prompt press <Ctrl>-<C> or <Ctrl>-<Break>. After issuing the END command, EDLIN renames your original file with the extension .BAK and saves the updated file with the file name and extension you gave when you started.

#### NOTE

You cannot update a file with an extension of .BAK because when you try to save the file, EDLIN will always save the original as .BAK losing your changes. If you need to edit such a file, rename it with another extension using the DOS REN command and start EDLIN again using the new file name.



## 5.2 Special EDLIN Commands

To edit your text files you can also use special editing keys.

The following table summarizes the commands, codes, and functions of the special editing keys. Descriptions of these keys follow the table.

**Table 5-1. Special EDLIN Keys.**

Key	Function
< F1 >	Copies one character from the template to the new line.
< F2 >	Copies all characters from the template to the new line, up to the character specified.
< F3 >	Copies all remaining characters in the template to the screen.
< Del >	Skips over a character.
< F4 >	Skips over the characters in the the template, up to the character specified.
< Esc >	Clears the current input and leaves the template unchanged.
< Ins >	Enters/exits insert mode.
< F5 >	Makes the new line the new template.

**<F1>**


---

**Purpose:** Copies one character from the template to the current line.

**Remarks:**

When you press <F1>, EDLIN copies one character from the template to the current line, and turns insert mode off

**Example:**

As an example of how to use <F1> with EDLIN, type the following line:

```
1:*Sharpe Office Supplies.
2.*_
```

At the beginning of the editing session, the cursor (shown by the underscore) is at the beginning of the line. When you press <F1>, EDLIN copies the first character to line 2 as shown here:

```
<F1>
2:*S_
```

Each time you press <F1> one more character appears.

```
<F1>
2:*Sh_
```

```
<F1>
2:*Sha_
```

```
<F1>
2:*Shar_
```

**< F2 >**

---

**Purpose:** Copies multiple characters up to a given character.

**Remarks:**

When you press < F2 >, EDLIN copies all the characters up a given character from the template to the current line. The given character is one that you type immediately after < F2 >.

EDLIN does not copy or display the given character on the screen, but it does copy and display the characters from the template up to the position of that character. If the template does not contain the character, EDLIN doesn't copy anything.

**Example:**

As an example of how to use the < F2 > key, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

When you press < F2 > followed by the letter C, EDLIN copies the characters up to the C in Office.

```
< F2 > c  
2:*Sharpe Offi_
```

**< F3 >**

---

**Purpose:** Copies the template to the current line.

**Remarks:**

When you press < F3 >, EDLIN copies the remaining characters in the template to the current line. No matter where the cursor is when you press < F3 >, EDLIN displays the rest of the line and leaves the cursor at the end of the line.

**Example:**

As an example of you to use < F3 > with EDLIN, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

When you press < F3 >, EDLIN copies the characters in the template to the line with the cursor:

```
< F3 >  
2:*Sharpe Office Supplies._
```

**< DEL >**

---

**Purpose:** Skips over one character in the template.

**Remarks:**

Each time you press < Del >, EDLIN skips over and does not copy the next character in the template.

**Example:**

As an example of how to use < Del >, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

Remember that at the beginning of the editing session, the cursor is at the beginning of the line. When you press < Del >, EDLIN skips over the first character, S:

```
< Del >  
2:*_
```

The cursor does not move as EDLIN changes the template. To see how much of the line has been skipped over, press < F3 >. This moves the cursor past the last character of the line:

```
< F3 >  
2:*harpe Office Supplies._
```

**< F4 >**

---

**Purpose:** Skips multiple characters in the template up to the specified character.

**Remarks:**

When you press < F4 >, EDLIN skips over all characters up to a given character in the template. EDLIN does not copy or display any of the character up to and including the given character. If the template does not contain that character, EDLIN does not skip any characters.

**Example:**

As an example of how to use < F4 >, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

When you press < F4 > followed by the letter c, EDLIN skips over all the characters in the template up to the c in the word *Office*.

```
< F4 > c  
2:*_
```

The cursor does not move as EDLIN changes the template. To see how much of the line has been skipped, press < F3 > to copy the template. This displays the rest of the line and moves the cursor to the end of the line.

```
< F3 >  
2:*ce Supplies._
```

**<ESC>**

---

**Purpose:** Quits input and clears the current line.

**Remarks:**

When you press <Esc>, EDLIN empties the current line and leaves the template unchanged. <Esc> also prints a backslash (\), return, and linefeed and turn insert mode off. The cursor is at the beginning of the line. If you press <F3>, EDLIN copies the template to the current line, making it appear as it was before you pressed <Esc>.

**Example:**

As an example of how to use <Esc>, type the following lines:

```
1:*Sharpe Office Supplies._  
2:*The World Leader_
```

To cancel the current line (line 2), press <Esc>. Notice that a backslash appears on line 2 to tell you it has been canceled:

```
<Esc>  
2:*The World Leader\
```

Press <Enter> to keep line 2, or to perform any other editing functions. Now if you press <F3>, EDLIN copies the original template to the line:

```
<F3>  
2:*Sharpe Office Supplies._
```

## < INS >

---

**Purpose:** Enters insert mode or replace mode.

**Remarks:**

When you start EDLIN, you are automatically in replace mode. The first time you press < Ins >, EDLIN enters Insert mode. In insert mode the cursor in the template does not move, but in the current line it moves as you insert each character. When you finish inserting characters and press < Ins > again, EDLIN re-enters Replace mode with the cursor at the same character in the template as when you entered the Insert mode.

In insert mode, EDLIN puts characters you type at the keyboard line into the template and in front of the cursor on the current line.

In replace mode, EDLIN replaces characters in the template and on the current line with characters you type on the keyboard.

**Example:**

As an example of how to use < Ins >, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

Remember that at the beginning of the editing session, the cursor is at the beginning of the line. press < F2 > and O:

```
< F2 > O  
2:*Sharpe _
```



Second, press <Ins> and type the word Automatic followed by a space:

```
<Ins> Automatic  
2:*Sharpe Automatic _
```

Now press <F3>. EDLIN copies the rest of the template to the line:

```
<F3>  
2:*Sharpe Automatic Office Supplies._
```

If you press <Enter> after entering insert mode, EDLIN does not copy the remainder of the template and ends the current line after the inserted text:

```
1:*Sharpe Office Supplies._  
2:*Sharpe Automatic  
3:*_
```

To exit insert mode and enter replace mode, simply press the <Ins> key again. Now all the characters you type will replace the characters in the template.

**<F5>**

---

**Purpose:** Creates a new template.

**Remarks:**

When you press <F5>, EDLIN copies the current line to the template and deletes the previous contents. Pressing <F5> also displays an @ (at), and out puts a return and a linefeed. When you press <F5>, EDLIN empties the current line and turns off Insert mode.

**NOTE**

<F5> performs the same function as <Esc> except that it changes the template, printing an @ sign instead of a backslash.

**Example:**

As an example of how to use <F5> with EDLIN, type the following line:

```
1:*Sharpe Office Supplies._  
2:*_
```

Remember that at the beginning of the editing session, the cursor is at the beginning of the line. Now type the following sequence of keys and words:

**<F2>** c

2:\*Sharpe Offi\_

**<Ins>** cial

2:\*Sharpe Official\_

**<Ins>** Sharpeware.

2:\*Sharpe Official Sharpeware.

The @ sign shows that this new line is now the new template. To add the word *Introducing:*, followed by a space, at the beginning of the line, press **<Ins>** and type the following sequence of keys and words:

**<Ins>** Introducing:

2:\*Introducing: \_

Press **<F3>** to insert the contents of the template.

**<F3>**

2:\*Introducing: Sharpe Official Sharpeware.\_

### 5.3 EDLIN Commands

This section describes the EDLIN commands, listed in Table 5-2.

**Table 5-2. EDLIN Commands.**

Name	Command	Function
Append	A	Appends lines.
Copy	C	Copies lines.
Delete	D	Deletes lines.
Edit	<i>lines</i>	Edits a line or lines.
End	E	Ends editing.
Insert	I	Inserts lines of text.
Move	M	Moves a range of text to specified line.
Page	P	Pages through a file 23 lines at a time.
Quit	Q	Quits the editing session without saving the file.
Replace	R	Replaces text.
Search	S	Searches for text.
Transfer	T	Transfers the contents of another file into the file being edited.
Write	W	Writes specified lines to a disk.

## A (Append)

---

**Purpose:** Adds a specified number of lines from your disk to the file being edited in memory.

**Format:** [n]A

where *n* specifies the number of lines you want read in memory. You need this command only if the file you want to edit is too large to fit into memory.

**Remarks:**

To edit the remainder of the file that will not fit into memory, you must write lines that you have already edited onto your disk. Then you can load unedited lines from your disk into memory by using the append (A) command. Refer to the write (W) command for information on how to write edited lines to your disk.

### NOTE

If you do not specify the number of lines to append, EDLIN adds lines to the available memory until it is three-quarters full, but does nothing if available memory is already three-quarters full.

After the A command reads the last line of the file into memory, EDLIN displays:

End of input file.

## C (Copy)

---

**Purpose:** Copies a range of lines to a specified line number, and when uses either the *count* option, copies this range as many times as you want.

**Format:** `[line], [line], LINE[count]C`

where:

The first and second *line* options specify the range of lines that you want to copy.

The third *line* option specifies the line before which EDLIN will place the copied lines.

*count* is the number of lines to be copied at one time.

**Remarks:**

You must not overlap the line numbers or you will get an "Entry error" message. For example the following command would result in an error message:

`3,20,25C`

**Example:**

As an example of how to use C with EDLIN, type the following line:

- 1: Sharpe Office Supplies.\_
- 2: The World Leader in Office Sharpeware
- 3: I.R. Sharpe, President
- 4:
- 5: "You oughta be Sharpe too."

You could copy this entire block of text by typing the following command:

1,5,6C

This command copies lines 1 through 5 and duplicates them one time, beginning on line 6. The result is:

- 1: Sharpe Office Supplies.\_
- 2: The World Leader in Office Sharpeware
- 3: I.R. Sharpe, President
- 4:
- 5: "You oughta be Sharpe too."
- 6: Sharpe Office Supplies.\_
- 7: The World Leader in Office Sharpeware
- 8: I.R. Sharpe, President
- 9:
- 10: "You oughta be Sharpe too."

If you wanted to place this text within other text, you would specify the third line option as the line you want the copied text to appear before.

For example, suppose you want to copy lines and insert them in the middle of the file. the command 7,10,5c would result in the following:

```
1: Sharpe Office Supplies.  
2: The World Leader in Office Sharpeware  
3: I.R. Sharpe, President  
4:  
5: The World Leader in Office Sharpeware  
6: I.R. Sharpe, President  
7:  
8: "You oughta be Sharpe too."  
9: "You oughta be Sharpe too."  
10: Sharpe Office Supplies.  
11: The World Leader in Office Sharpeware  
12: I.R. Sharpe, President  
13:  
14: "You oughta be Sharpe too."
```



## D (Delete)

---

**Purpose:** Deletes a specified range of lines in a file.

**Format:** [*line*][*,line*]D

where *line,line* specifies the range of lines to be deleted

**Remarks:**

If you omit the first *line* option, EDLIN defaults to the current line.  
If you omit the second *line* option, then EDLIN deletes just the first *line*.

**Example:**

Suppose the following file exists and ready to edit:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
11: to sharpen metal objects.
12:
13: We hope that you will be more
14: careful with electrical appliances
15: in the future.
16:
17: Sincerely,
18:
19: *I.R. Sharpe, President
```

But now, say you decide not to warn Mr. Dimm about being careful. To delete lines 12 through 15, type:

```
12,15D
```

The result of this command is:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: *I.R. Sharpe, President
```

To close up the space you might decide to delete line 7. Type:

7D

The result would be:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7: *As a result of your accident, we
8: are redesigning our manual
9: to warn our customers against trying
10: to sharpen metal objects.
11:
12: Sincerely,
13:
14: I.R. Sharpe, President
```

Suppose you just want a quick message that doesn't take responsibility for the accident. To delete a range of lines beginning with the current line, line 7, through line 11, type:

,11d

The result would be:

1: Dear Mr. Dimm:  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: Sincerely,  
9:  
10:\*I.R. Sharpe, President

## Edit

---

**Purpose:** Edits a line of text.

**Format:** [line]

**Remarks:**

The line option specifies the line of text you want to edit. When you type a line number as a command, EDLIN displays the line number and the text on that line. Then, on the line below, EDLIN reprints the line number. You can retype the line or use the editing keys to edit. The existing text of the line serves as the template until you press <Enter>.

If you do not type a line number and only press <Enter>, EDLIN edits the line after the current line.

If you do not need to change the current line and if the cursor is at the beginning or end of a line, you can simply press <Enter> to accept the line.

### NOTE

If you press <Enter> while the cursor is in the middle of a line, EDLIN deletes the remainder of the line

**Example:**

Suppose that the following file exists and is ready to edit:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our Automatic
6: Pencil Sharpener.
```

In line 5, say you wanted to insert the product's name, X-1000. To edit line 5, type the number 5. EDLIN then displays the contents of the line with the cursor below the line:

```
5:*shock from our Automatic
5:* _
```

Use <F2> to skip to the A in the word *Automatic* and type:

```
<F2> A <Ins> X-1000
5:*shock from our X-1000
```

```
<F3> <Enter>
5:*shock from our X-1000 Automatic
```

At the EDLIN prompt, type L to see the file:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
```

## E (End)

---

**Purpose:** Ends the editing session.

**Format:** E

**Remarks:**

The E (end) command saves the edited file on your disk, renames the original input file with a .BAK extension and then exits EDLIN. If you created the file during the current editing session, EDLIN does not create a backup file.

The E command takes no options. This means that you must select the drive that you save the file on when you start EDLIN. If you don't, EDLIN saves the file on the disk in the default drive. You can still copy the file to a different drive by using the MS-DOS command COPY.

Before using the E command to save your file, make sure that the disk contains enough free space for the entire file. If it doesn't, EDLIN may not be able to write the entire file to the disk. The edited file will be lost, even though EDLIN may have saved part of it on the disk.

**Example:**

To end your editing session, type E and press <Enter>. After EDLIN processes the E command, the screen displays the DOS prompt.

## I (Insert)

---

**Purpose:** Inserts text immediately before the specified line.

**Format:** [*line*] I

**Remarks:**

If you are creating a new file, you must type the I (insert) command before you can insert a new line of text. Text begins on line one, and each time you press the return key the next line number appears automatically.

EDLIN remains in insert mode until you press <Ctrl>-<C>. When you finish the insertion and exit insert mode, the line immediately following the inserted lines becomes the current line. EDLIN automatically increments the line numbers that follow the inserted section by the number of lines that you inserted.

If you do not specify *line*, the default is the current line number and EDLIN inserts the lines before the current line. If a *line* number is larger than the last line number of the file, or if you specify a pound sign (#) as *line*, EDLIN appends the inserted lines to the end of the file. In this case the last line that you inserted becomes the current line.



**Examples:**

Suppose the following file exists and is ready to edit:

```
1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: Sincerely,
9:
10: I.R. Sharpe, President
```

Since this letter doesn't really offer any compensation for the accident, you might decide to add a brief note about a small gift that you are enclosing for Mr. Dimm. To insert text before line 8, type:

```
8I
```

The result is:

```
8:* _
```

Now type following lines which will begin on line 7:

```
8:*As a result of you accident
```

Press < **Enter** > at the end of each line and continue typing the next line.

```
9:*we are redesigning our manual to
10:*warn our customers against trying
11:*sharpen metal objects.
```

To end the insertion, press <Ctrl>-<C> on the next line and type L. The result is:

```

1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
11: to sharpen metal objects.
12:*Sincerely,
13:
14: I.R. Sharpe, President

```

To insert a blank line immediately before the current line (line 12) type:

```

I

```

The result is:

```

12 :*_

```

Insert a blank line by pressing <Enter> and end the Insertion by pressing <Ctrl>-<C> on the next line. Then to list the file and see the result, type L.

1: Dear Mr. Dimm:  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: As a result of your accident, we  
9: are redesigning our manual  
10: to warn our customers against trying  
11: to sharpen metal objects.  
12:  
13: \*Sincerely,  
14:  
15: I.R. Sharpe, President

To append new lines to the end of the file, type:

16l

Now type the following new lines:

16: Sharpe Office Supplies  
17: The World Leader in Office Sharpeware  
18: Our Motto: "You ought be Sharpe too."

To get out of insert mode press <Ctrl> - <C> on line 19. The new lines will appear at the end of all previous lines in the file. Now list all the lines by typing:

1,23L

The result is:

1: Dear Mr. Dimm:  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: As a result of your accident, we  
9: are redesigning our manual  
10: to warn our customers against trying  
11: to sharpen metal objects.  
12:  
13: Sincerely,  
14:  
15: \*I.R. Sharpe, President  
16: Sharpe Office Supplies  
17: The World Leader in Office Sharpeware  
18: Our Motto: "You ought be Sharpe too."

## L (List)

---

**Purpose:** Lists a range of lines, including the two lines specified.

**Format:** [*line*][, *line*]L

**Remarks:**

A capital L has been use here to avoid confusion with the number one. A small l would work just as well.

EDLIN provides default values if you do not type elther option. If you do not type the first *line* option, as in the following command, the display will start 11 lines before the current and end with the specified *line*.

,*line*L

The beginning comma is needed to show that omitted the first *line* option.

### NOTE

If the specified *line* is more than 11 lines before the current line, the display will be the same as if you omitted both options.

If you type L, EDLIN displays 23 lines -- the 11 lines before the current line, the current line and the 11 lines after the current line. If there are less than 11 lines before the current line, EDLIN displays more than 11 lines after the current line up to a total of 23 lines.

**Examples:**

Suppose the following file exists and is ready to edit:

```

1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
:
:
:
15: We also intend to attach a metal
16: detector to future models.
:
:
:
26:*I.R. Sharpe, President
27: Sharpe Office Supplies
28: The World Leader in Office Sharpeware
29: Our Motto: "You ought be Sharpe too."
```

To list a range of lines without referring to the current line, type:

2,6L

The result is:

```

2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
```

To list a range of lines beginning with the current line, type:

16,L or ,l

The result is:

16: detector to future models.  
:  
:  
:  
26:\*I.R. Sharpe, President  
27: Sharpe Office Supplies  
28: The World Leader In Office Sharpeware  
29: Our Motto: "You ought be Sharpe too."

To list a range of 23 lines centered around the current line, type:

L

The result is:

5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
:  
:  
:  
15: We also intend to attach a metal  
16: detector to future models.  
:  
:  
:  
26:\*I.R. Sharpe, President  
27: Sharpe Office Supplies

## M (Move)

---

**Purpose:** Moves a range of text to the specified line.

**Format:** [line],+line,lineM

**Remarks:**

The M (move) command lets you move a block of text to another location in a file. The first and second *line* options specify the range of lines that you want to move. The third *line* option specifies the line you want to move the first line in the range to.

EDLIN automatically renumbers the lines after it moves them. For example, the following command moves the text from the current line -- plus 25 lines -- to line 100.

,+25,100M

If the line numbers you specify overlap, EDLIN displays an "Entry error" message.



**Example:**

Suppose the following example exists and is ready to edit.

1: Dear Mr. Dimm:  
2:  
3: I was sorry to hear of your recent  
4: hospitalization due to electrical  
5: shock from our X-1000 Automatic  
6: Pencil Sharpener.  
7:  
8: As a result of your accident, we  
9: are redesigning our manual  
10: to warn our customers against trying  
11: to sharpen metal objects.  
12:  
13: Sincerely,  
14:  
15: \*I.R. Sharpe, President  
16: Sharpe Office Supplies  
17: The World Leader in Office Sharpeware  
18: Our Motto: "You ought be Sharpe too."

What if you prefer to have the motto at the start of the letter? If so, you could move lines 16 -- 18 to line 1 by typing:

16,18,1M

The result of this command is:

1: Sharpe Office Supplies  
2: The World Leader in Office Sharpeware  
3: Our Motto: "You ought be Sharpe too."  
4: Dear Mr. Dimm:  
5:  
6: I was sorry to hear of your recent  
7: hospitalization due to electrical  
8: shock from our X-1000 Automatic  
9: Pencil Sharpener.  
10:  
11: As a result of your accident, we  
12: are redesigning our manual  
13: to warn our customers against trying  
14: to sharpen metal objects.  
15:  
16: Sincerely,  
18:  
19: I.R. Sharpe, President

## P (Page)

---

**Purpose:** Displays a file on page (23 lines) at a time.

**Format:** [*line*][, *line*]P

**Remarks:**

The first *line* option specifies the line where EDLIN starts displaying. The second *line* option specifies how many lines on each page. If you do not type the first *line* EDLIN starts the page at the line after the current line. If you do not type the second *line* option, EDLIN lists 23 lines on each page

## Q (Quit)

---

**Purpose:** Quits the editing session, does not save any editing changes, and exits to the DOS prompt.

**Format:** Q

**Remarks:**

This command is useful if you don't want to make any changes to a file. If you use the **Q** (quit) command, EDLIN prompts you to make sure you don't want to save the changes. If you do want to save your changes, you should refer to the **E** (end) command for information about the *.bak* file.

**Note**

When you exit EDLIN, it erases any previous of the file that has a *.bak* extension. If you reply **Y** to the "Abort edit (Y/N)?" message, EDLIN deletes your previous backup copy.

**Example:**

The following example shows how to quit EDLIN without saving your changes. First, type **Q** at the EDLIN prompt. At the "Abort edit (Y/N)?" message type **Y**.

## R (Replace)

---

**Purpose:** Replaces all occurrences of a string of text in a range with a different string of text.

**Format:** `[line][,line][?]Rtext1  
<Ctrl>-<Z> text2`

where:

*line, line text1* specify the range of lines the R (replace) command uses.

*text2* specifies replacement text.

### Remarks:

If a line contains two or more replacements, it is displayed once for each change. When EDLIN has made all the change, the R command ends and the asterisk prompt re-appears.

If you want to replace one string of text with another, you num = st separate the two with a <Ctrl>-<Z>. You can end the second string by pressing <Enter>.

If you omit the first *line* option, EDLIN uses the line after the current line, by default. The default for the second *line* option is #. Remember that # refers to the line after the last line of the file.

If you end *text1* with a <Ctrl>-<Z> and do not specify *text2*, EDLIN assumes you want blank spaces for *text2*. For example, suppose you want to delete all occurrences of the word *clients* from your file. To do this you could simply type:

```
RCLIENTS <Ctrl>-<Z> <Enter>
```

The next command replaces *clients* with the previous *text2*. Type:

```
RCLIENTS
```

The following command makes the previous *text1* become the previous *text2*. Type:

```
R
```

Note that "previous:" refers to an earlier string of text specified in an S or R command.

If you specify a question mark (?) in the the R command, EDLIN stops at each line with text that matches *text1*, displays the line with *text2* and displays the prompt "O.K.?" If you press Y or <Enter>, *text2* replaces *text1*, and EDLIN looks for the next occurrence of *text1*.

**Example:**

Suppose the following file exists and is ready for editing.

```

1: Dear Mr. Dimm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: *I.R. Sharpe, President

```

Now suppose that lines 5 through 10 you want to replace all occurrences of the word *our* with the word *the*. To do this, type:

```
5,10 our <Ctrl>-<Z> the <Enter>
```

The result is:

```

5: shock from the X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn the customers against trying

```

In the previous example, some unwanted changes occurred. To avoid these and to confirm each replacement, you can use the same file with a slightly different command.

In the next example you will see how to replace only certain occurrences of *our* with *the*. At the EDLIN prompt, type the following sequence of keys and words, and press <Enter>.

1,15? rour <Ctrl>-<Z> the

The result is:

```

5: shock from the X-1000 Automatic
O.K.? y
8: As a result of the accident, we
O.K.? n
9: are redesigning our manual
O.K.? Y
10: to warn the customers against trying
O.K.? n
*
_

```

Type the list command, L, to see the result of all these changes:

```

.
.
5: shock from our X-1000 Automatic
.
.
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
.
.

```



## S (Search)

---

**Purpose:** Searches a range of lines for a string of text.

**Format:** [line][,line] [?] Stext

where:

*line, line* specifies range of lines for EDLIN to search.

*text* specifies the search text.

### Remarks:

EDLIN displays the first line that matches the string. That line becomes the current line. Unless you type the question mark (?) option, the S (search) command ends when it finds the first match. If EDLIN cannot find a line with a match, it displays the message "Not found."

If you include the question mark option, EDLIN displays the first line with matching text and prompts you with the message "O.K.?" If you press either Y (yes) or <Enter>, this line becomes the current line and the search ends. If you press any other key, the search continues until another match is found or until all lines have been searched. The search ends when EDLIN displays the "Not found" message.

If you don't type the first line number, EDLIN defaults to the line after the current line. If you don't type the second line number, it defaults to the last line after the last line of text.

If you omit the *text* option, EDLIN uses the text from the previous S or R command. If this is the first S or R command you have used during the current session and you have not specified a search string, the search ends immediately.

**Example:**

Suppose the following file exists and is ready for editing:

```

1: Dear Mr. DImm:
2:
3: I was sorry to hear of your recent
4: hospitalization due to electrical
5: shock from our X-1000 Automatic
6: Pencil Sharpener.
7:
8: As a result of your accident, we
9: are redesigning our manual
10: to warn our customers against trying
11: to sharpen metal objects.
12:
13: Sincerely,
14:
15: *I.R. Sharpe, President

```

To search for the first occurrence of the word *to*, type:

**2,12 Sto**

EDLIN displays the following:

```

3: I was sorry to hear of your recent

```

To get to the *to* in line 4, modify the S command by pressing  
**<Del>-<F3>-<Enter>**. To continue the search, type:

**S <Enter>**

The next line containing the search text displays:

4: hospitalization due to electrical

To search through several lines until the desired occurrence is found, type:

**1,12 ? Sto**

The result is:

3: I was sorry to hear of your recent  
O.K.? \_

Type **N** to continue the search.

4: hospitalization due to electrical  
O.K.? \_

Now press **Y** to terminate the search:

**Messages:**

Not found

Displays if EDLIN cannot find a match.

## T (Transfer)

---

**Purpose:** Inserts, at a specific line number, the contents of another file into the file you are currently working on.

**Format:** [line]Tfile name

where:

*line* specifies where to insert the new file in your current file.

*file name* is the name of the new file.

**Remarks:**

The T (transfer) command puts the contents of one file into another, on into the text you are typing. If you omit the line number, EDLIN inserts the text, beginning on the current line.

**Example:**

To copy a file named *IRSHARPE.MEM* to line 12 of the file you are editing, use the following command:

```
12 T IRSHARPE.MEM
```

## W (Write)

---

**Purpose:** Writes a specific number of lines to a disk

**Format:** [n]W

**Remarks:**

The *n* option specifies the number of lines that you want to write to the disk. You need this command only if the file you are editing is too large to fit into memory. When you start EDLIN, it reads lines from your file until memory is three-quarters full.

To edit the remainder of your file, you must write the edited lines in memory to your disk. Then you can load additional unedited lines from your disk into memory by using the A (append) command.

### NOTE

If you do not specify the number of lines for EDLIN to write, it writes lines until memory is three-quarters full. But it does not write any lines to your disk until memory is more than three-quarters full. Also, EDLIN remembers all of the lines so that the first remaining line becomes line 1.

LINK



---

The Microsoft Object Linker (LINK) creates executable programs from object files generated by the Microsoft Macro Assembler (MASM) or by compilers for high-level languages, such as C or Pascal. The linker copies the resulting program to an executable (.exe) output file. You can then run the program by typing the file's name on the disk operating system (DOS) command line.

To use LINK, you must create one or more object files, then submit these files, along with any required library files, to the linker for processing. LINK combines code and data in the object files and searches the named libraries to resolve external references to routines and variables. It then copies a relocatable execution image and the relocation information to the executable file. Using the relocation information, DOS can load the executable image at any convenient memory location and then run it. LINK can process programs that contain up to one megabyte (MB) of code and data.

This section explains how to use the linker to create executable programs. It also defines each of the options you can use in a LINK command line to control the linking process. Finally, it explains how LINK creates programs.



## 6.1 Starting and Using LINK

This section explains three methods for starting and using the linker to create executable programs. These methods let you use LINK by answering a series of prompts, by typing a DOS command or by using a response file. The three methods can also be mixed.

Once you start LINK, it will either process the files you supply or prompt you for additional files. Also note that you can stop the linker at any time by pressing <Ctrl>-<C>.

### 6.1.1 Using Prompts to Specify LINK Files

When you type the command, LINK, at the DOS prompt, the linker prompts you for the information it needs.

1. At the DOS prompt, type:

**LINK <Enter>**

LINK prompts you for the object files you wish to link by displaying the following message:

Object Modules [.OBJ]:

2. Type the name or names of the object files you wish to link. If you do not supply extensions for these files, LINK supplies .OBJ by default. If you have more than one name, make sure you separate each with spaces or a plus sign (+). If you have more names than can fit on one line, type a plus sign as the last character on the line and press <Enter>. LINK then prompts you for additional files.

Once you have given all your object file names, press <Enter>. The linker displays the following prompt:

Run File [file name.exe]

3. Type the name of the executable file you wish to create, and press <Enter>. If you do not give an extension, LINK supplies .EXE by default. If you want LINK to supply a default executable file name, press <Enter>. The file name will then be the same as the first object file, but with the extension .EXE. Once you have pressed <Enter>, LINK displays the following prompt:

List File [NUL.MAP]:

4. Type the name of the map file you wish to create, then press **<Enter>**. If you do not supply a file name extension, the linker use *.map* by default. If you don't want a map file, don't type a file name. Press **<Enter>**.

Once you have pressed **<Enter>**, LINK displays the following prompt:

Libraries [.LIB]:

5. Type the names of any library files containing routines or variables referenced but not defined in your program. If you have more than one name, make sure the names are separated by spaces or plus signs (+). If you don't supply file name extensions, LINK uses *.LIB* by default. If you have more names than can fit on one line, type a plus sign as the last character on the line and press **<Enter>**. LINK then prompts you for additional file names.

After entering all names press **<Enter>**. If you don't want to search any libraries, don't type any names; just press **<Enter>**.

LINK now creates the executable file.

When entering file names, you must supply a path name for any file that is not in the current drive and directory. You can use LINK options by typing them after the file name at any prompt. If the linker cannot find an object file, it displays a message and waits for you to change disks, if necessary.

At any prompt, you can type the rest of the file names by using the command line format described in the next section. For example, you can choose the default responses for all remaining prompts by typing a semicolon (;) after any prompt, or you can type commas (,) to indicate several files. When the linker encounters a semicolon, it immediately chooses the default responses and processes the remaining files without displaying any more prompts.

#### Example:

The following example links the object modules *moda.obj*, *modb.obj*, *modc.obj* and *startup.obj*; searches the library file *math.lib* on drive B of the *\lib* directory for routines and data used in the program; and creates an executable file named *moda.exe*, and a map file named *abc.map*.

The */pause* option in the Object Modules prompt line then causes LINK to pause while you change disks, after which the linker creates the executable file.

#### LINK

```
Object Modules [.OBJ]: moda + modb +
Object Modules [.OBJ]: modc + startup/PAUSE
Run File [moda.exe]:
List File [NUL.MAP]: abc
Libraries [.LIB]: b:\lib\math
```

### 6.1.2 Using a Command Line to Specify LINK Files

You can create an executable program by typing LINK, followed by the names of the files you wish to process.

**Format:** LINK object files [, [executable file] [, [map file] [, library file]]]] [options][:]

where:

*object files* are the name or names of object files that you want to link together. The files must have been created using MASM or a high-level language compiler. The linker requires at least one object file. If you do not supply an extension, LINK provides the extension *.obj*.

*executable file* is an optional placeholder for the name you wish to give the executable file that LINK will create. If you do not supply an *executable file*, LINK creates a file name by using the name of the first object file in the command line and appending it with an *.exe* extension.

*map file* is the name of the file that receives the map listing. If you do not supply an extension, the linker provides the extension *.map*. If you specify the */map* or */line* numbers option, the linker creates a map file even if you don't specify one in your command line.

*library files* includes the name or names of the libraries containing routines that you wish to link to create a program. If you do not supply an extension, LINK supplies the extension *.lib*.

*options* control the operations of LINK. You can use any of the options listed in Section 6.2. You can specify options anywhere on the command line.

If you do not specify a drive or directory for a file LINK assumes the file is on the current drive and directory. You cannot specify the drive or directory for the *object file* and expect LINK to supply the same drive and directory for other files. Instead, you must give the location of each file specifically.

If you type the comma after the object file, supplies the default name for the *executable file* and suppresses the *map file* and *library files*. You can also use a semicolon (;) anywhere after the object file to terminate the command line.

If you do not supply all file names in the command line and do not end the command line with a semicolon, the linker prompts you for additional files, using the prompts described previously in section 6.1.1. If you file more than one object file or library file, you must separate the name by spaces or a plus sign (+).

If you do not specify a drive or directory for a file, LINK assumes the file is on the the current drive and directory. So, you cannot specify the drive or directory for the *object file* and expect LINK to supply the same drive and directory for other files. Instead, you must give the location of each file specifically.

When linking modules produced by a high-level language compiler that supports overlays, you must specify overlay modules by putting them in parentheses. Since MASM has no overlay manager, you can specify overlays only for object files linked with the run-time library of a language compiler that supports overlays.

**Example:**

The first example below uses the object file *file.obj* to create the executable file *file.exe*. LINK searches the *file.lib* library for routines and variables used within the program. It also creates a file called *file.map*, which contains a list of the program's segments and groups:

```
link file.obj,file.exe,file.map,routine
.lib
```

The first example is equivalent to the following line:jp

```
link file,,,routine
```

The second example uses the two object files, *startup.obj* and *file.obj*, on the current drive to create an executable file named *file.exe* on drive B. LINK creates a map file on the *\map* directory of the current drive, but does not search any libraries:

```
link startup + file,b:file,\map\file
```

The final example links the object modules *moda.obj*, *modb.obj*, *modc.obj*, and *startup.obj*:

```
link moda modb modc startup/pause,,abc,b:\lib\math
```

The linker searches through the library file *math.lib* in the *\lib* directory on drive B for routines and data used in the program. It then creates an executable file named *moda.exe* and a map file named *abc.map*.

The pause option in the command line causes the linker to pause and ask you to change disk before it creates the executable file.

### 6.1.3 Using a Response File to Specify LINK Files

You can create a program by listing, in a response file, the name of all the files to be processed, and by giving the name of the response file on the LINK command line. The simplest way to use a response file is with a command line of the following form:

LINK @file name

You can also specify a response file at any prompt or at any position in a command line. The input from the response file is treated exactly as though you had type it at the LINK prompts or in a command line. However, any return/linefeed combinations in the file are treated the same as if you had press <Enter> in, response to a prompt, or type a comma in a command line.

When you specify a response file, remember that the file name must be the name of the response file, and that you must precede it by an "at" sign (@). If the file is in another directory or on another disk drive, you must provide a path name.

You can name the response file anything you like. The file content has the following general form:

*object files*  
*[executable file]*  
*[map file]*  
*[library files]*

You can omit any elements that have already been provided at prompts or with a partial command line.



You must place each group of file names on a separate line. If you have more names than can fit on one line, you can continue the name on the next line by typing a plus sign as the last character in the current line and pressing <Enter>. If you do not supply a file name for a group, you must leave an empty line. You can give options on any line.

You can place a semicolon (;) on any line in the response file. When LINK encounters the semicolon, it automatically supplies default file names for all files you have not yet named on the response file. The remainder of the response file is ignored.

When you create a program with a response file, the linker displays each response from your response file on the screen in the form of prompts. If the response file does not contain names for required files, prompts you for the missing names and waits for you to enter responses.

### Example:

The following response file tells LINK to link the four object modules, *moda*, *modb*, *modc* and *startup*. Then, before producing the executable file *moda.exe*, it tells LINK to pause to let you swap disks. Finally, the linker creates a map file *abc.map* and searches the *math.lib* library in the \lib directory of drive B:.

```
moda modb modc startup /pause
abc
b:\lib\math
```

The following procedure combines all three methods of supplying file names. Assume you have a response file called *library* that contains one line:

```
lib1 + lib2 + lib3 + lib4
```

Now start LINK with a partial command line:

```
link object1 object2
```

LINK takes *object1.obj*, and *object2.obj* as its object files and prompts for the next file:

```
Run file [object1.exe]: exe  
list file [nul.map] :  
libraries [.lib]: @library
```

You include the name *exec* so the linker will name the executable file *exec.exe*. You then press <Enter> to indicate that no map file is desired, and you enter *@library* so the linker will read in the response file containing the four library file names.

#### 6.1.4 Using Search Paths with Libraries

You can direct LINK to search directories and disk drive for the libraries you have named in a command by either specifying one or more search paths with the library names, or by assigning the search paths to the environment variable LIB before you enter LINK.

A search path is the path of directory or drive name. You type search paths along with library names on the LINK command line or in response to the Libraries prompt. You can also specify up to 16 search paths and assign them to the LIB environment variable by using the SET command. In the latter case, you must separate the search paths by semicolons.

If you include a drive or directory name in the file name for a library in the LINK command line, the linker searches there only. If you don't give a drive or directory name, LINK searches for library files in the following order.

1. The linker searches the current drive and directory.
2. If the library is not found and one or more search paths have been given in the command line, the linker searches the specified search paths in the order in which you gave them.
3. If the library is still not found and you have set a search path by using the LIB environment variable, the linker searches there.
4. If the library is still not found, LINK prints an error message.

### Example:

In the first example, the linker searches only the `\altdlib` directory on drive A to find the `math.lib` library. To find `common.lib`, it will search the current directory on the current drive, the current directory on drive B, and finally the `\lib` directory on drive D:

```
link file,,file, a:\altdlib\math.lib + common + b:
+ d:\lib\
```

In the second example, LINK searches the current directory, the `\lib` directory on drive C, and the `\system\lib` directory on drive U to find the libraries `math.lib` and `common.lib`:

```
set lib = c:\lib;u:\system\lib
link file,,file.map,math + common
```

### 6.1.5 The Map File

The map file lists the names, load addresses, and lengths of all segments in a program. It also lists the names and load address of any groups in the program, the program start address, and messages about any errors it may have encountered. If the /map option is used in the LINK command line, the map file lists the names and load addresses of all public symbols.

Segment information has the general form shown in this example:

Start	Stop	Length	Name	Class
00000h	0172Ch	0172Dh	TEXT	CODE
01730h	01E19h	006EAh	DATA	DATA

The Start and Stop columns show the 20-bit addressed (in hexadecimal) of the first and last byte in each segment. These addresses are relative to the beginning of the load module, which is assumed to be address 00000h. The operating system chooses its own starting address once the program is actually loaded. The column gives the name of the segment and the Class column gives the segment's address.

Group Information has the following form:

Origin	Group
0000:0	IGROUP
0173:0	DGROUP

In this example, IGROUP is the name of the code (instruction) group and DGROUP is the name of the data group.

At the end of the listing, file, the linker gives you the address of the program entry point.

If you specify the /map option in the LINK command line, the linker adds a public-symbol list to the map file. The symbols are presented twice: once in alphabetical order then in the order of their addresses. The list has the general form shown in the following example:

Address	Publics by Name
0000:1567	BRK
0000:1696	CHMOD
0000:01DB	CHKSTK
0000:131C	CLEARERR
0173:0035	FAC

Address	Publics by Value
0000:01DB	CHKSTK
0000:131C	CLEARERR
0000:1567	BRK
0000:1696	CHMOD
0000:0035	FAC

The addresses of the public symbols are in segment:offset format. They show the location of the symbol relative to the beginning of the load module, which is assumed to be a address 0000:0000h.

When the /high and /dsallocate options are used and the program's code and data combined do not exceed 64 kilobytes (KB), the map file may show symbols that have unusually large segment address. These addresses indicate a symbol whose location is below the actual start of the program code data.

For example, the following symbol entry shows that TEMPLATE is located below the start of the program:

FFFO:0A20:0A20	TEMPLATE
----------------	----------

Note that the 20-bit address of TEMPLATE is 00920h.

### 6.1.6 The Temporary Disk File

LINK normally uses available memory for the link session. If it runs out of available memory, it creates a temporary disk file named *vm.tmp* in the current working directory. When the linker creates this file, it displays the following message:

VM.TMP has been created.  
Do not change diskette in drive x:

After this message appears, you must not remove the disk from the drive specified by x until the link session ends. the /pause option cannot be used if a temporary file is created. After LINK has created the executable file, it deletes the temporary file automatically.

#### NOTE

Do not use the *vm.tmp* file name for your own files, since when the linker creates the temporary file, it destroys any previous file that has the same name.

## 6.2 The LINK Options

The linker options specify and control the tasks that LINK performs. All options begin with the linker-option character, which is a slash (/). You can use the following options anywhere on a LINK command line.

**Table 6-1. LINK Options.**

Option Name	Action
/help	Shows the list of options.
/pause	Pauses during linking.
/exepack	Packs executable files.
/map	Creates public-symbol map.
/line numbers	Preserves case sensitivity for a program.
/nodefaultlibrarysearch	Overrides default libraries.
/stack allocation space.	Sets maximum
/high	Sets a high load address for a program.
/dsallocate	Allocates a data group.

Table 6-1. Continued.

/nogroupassociation	Sets a group association override.
/overlayinterrupt	Sets an overlay interrupt.
/segments	Sets a maximum number of segments.
/dosseg	Specifies DOS segment ordering.

You can abbreviate an option name as long as your abbreviation contains enough letters to distinguish the specified option from other options. Minimum abbreviations are listed for each option.

Many of the LINK options set values in the DOS program header.



### 6.2.1 Viewing the Options List

**Format:**            /help

**Remarks:**

The /help option causes LINK to write a list of the available options to the screen. If you ever need a reminder of the available options, you may find this list convenient. You should not give a file name when using the /help option.

**Abbreviation:** /he

**Example:**

```
link /help
```

### 6.2.2 Pausing to Change Disks

**Format:**            /pause

**Remarks:**

The /pause option causes LINK to pause before writing the executable file to disk so that you can swap disks before the linker writes the executable (.exe) file to disk.

If you specify the /pause switch, the linker displays the following message before creating the run file:

About to generate .EXE file  
Change diskette in drive x: and press <Enter>

Note that x: is the proper drive name. This message appears after the linker has read data from the object files and library files, and after it has written data to the map file, if you specified one. LINK resumes processing when you press <Enter>, and after it writes the executable file to disk, it displays the following message:

Please replace original diskette  
in drive x: and press <Enter>

**Abbreviation:** /p

#### NOTE

Do not remove the disk used for the *vm.tmp* file, if such a file has been created. If the temporary disk message appears when you have specified the /pause option, you should press <Ctrl> - <C> to terminate the LINK session. Rearrange your files so the temporary file and the executable file can be written to the same disk, then try again.

#### Example:

The following command causes the linker to pause just before creating the executable file *file.exe*. After creating this file, LINK pauses again to let you replace the original disk:

```
LINK FILE/PAUSE,FILE,,\LIB\MATH
```

### 6.2.3 Packing Executable Files

**Format:**            /exepack

**Remarks:**

The /exepack option directs LINK to remove sequences of repeated bytes (typically nulls) and optimize the load-time relocation table before creating the executable file. Executable files linked with the /exepack option may be smaller, and thus load faster than files linked without the option. However, the Symbolic Debug Utility (SYMDEB) cannot be used with packed files.

The /exepack option does not always save a significant amount of disk space (in some cases it may even increase file size). Programs that have a large number of load-time relocations (more than 500) and long streams of repeated characters will usually be shorter if packed. If you are not sure of your program meets these condition, try linking both ways and compare the results.

**Abbreviation:**   /e

**Example:**

This example creates a packed version of the file *program.exe*:

```
LINK PROGRAM /E;
```

## 6.2.4 Producing a Public-Symbol Map

**Format:**            /map

**Remarks:**

The /map option causes LINK to produce a listing of all public symbols declared in your program. This list is copied to the map file that LINK creates. The /map option is required if you want to use SYMDEB for symbolic debugging.

**Abbreviation:** /m

### NOTE

If you do not specify a map file in a LINK command, you can use the /map option to force the linker to create one. LINK gives the forced map file the same file name as the first object file specified in the command. It also adds the default extension .map

**Example:**

The following command creates a map of all public symbols in the file *file.obj*:

```
LINK FILE, ,/MAP;
```

### 6.2.5 Copying Line Numbers to the Map File

**Format:**            /line numbers

**Remarks:**

The /line numbers option directs the linker to copy the starting address of each program source line to a map file. The starting address is actually the address of the first instruction that corresponds to the source line. You can use the MAPSYM program to copy the line number data to a symbol file, which can then be used by SYMDEB.

The linker copies the line number data only if you give a map file name in the LINK command line, and only if the given object file has line number information. Line numbering is available in some high level languages compilers.

MASM does not copy line number information to the object file. If an object file has no line number information, LINK ignores the /line number option.

**Abbreviation:** /li

**Example:**

This example causes the line number information in the object file *file.obj* to be copied to the map file *file.map*.

```
LINK FILE/LINE NUMBERS, , EM+SLIBFP
```

### 6.2.6 Preserving Lowercase

**Format:**            /noignorecase

**Remarks:**

The /noignorecase option directs LINK to treat uppercase and lowercase in symbol names as distinct letters. Normally, LINK considers upper and lowercase letters to be identical, treating the words "TWO" and "two" identical. When you use the /noignorecase option, the linker treats "TWO" and "two" as different symbols.

If you are linking modules created with MASM to modules created with a case-sensitive language, such as C, make sure public symbols have the same sensitivity in both modules. For example, you could make all variables in C distinctive by spelling, regardless of case, and then link without the /noignorecase option. Another alternative would be to use the /ml or mx options to make public variables in MASM case-sensitive. Then link with the /noignorecase option.

**Abbreviation:** /noi

**Example:**

The following command causes the linker to treat uppercase and lowercase letters in symbol names as distinct letters. The object file *FILE.OBJ* is linked with routines from the standard C language library *\Slbc.lib* located in the *\lib* directory. The C language expects upper and lowercase letters to be treated distinctly:

```
LINK FILE1+FILE2/NOI,,,EM+MLIBFP
```

### 6.2.7 Ignoring Default Libraries

**Format:**            /nodefaultlibrarysearch

**Remarks:**

The /nodefaultlibrarysearch option directs the linker to ignore any library names it may find in an object file. A high-level language compiler may add a library name to an object file to ensure a default set of libraries is linked with the program. Using this option overrides these default libraries and lets you explicitly name the libraries you want by including them on the LINK command line.

**Abbreviation:** /nod

**Example:**

The following example links the object files, *startup.obj*, and *file.obj*, with routines from the libraries, *em*, *slibfp*, and *slibc*. Any default libraries that may have been named in *startup.obj* or *file.obj* are ignored:

```
LINK STARTUP+FILE/NOD, ,EM+SLIBFP+SLIBC
```



### 6.2.8 Setting The Stack Size

**Format:**            /stack:size

**Remarks:**

The /stack option sets the program stack to the number of bytes given by size. The linker usually calculates a program's stack size automatically, basing it on the size of any stack segments given in the object files. If you do use the /stack option, the linker uses the value you type in place of any value it may have calculated.

The size can be any positive number between 1 and 65535. This value can be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero, and hexadecimal numbers must begin with a leading zero followed by a lowercase x. For example 0x1B.

Using the EXEMOD utility, you can also change the stack size after linking.

**Abbreviation:** /st

**Example:**

The first example sets the stack size 512 bytes:

```
LINK FILE/STACK:512, , ;
```

The second example sets the stack size to 255 (0FFh) bytes:

```
LINK MODA+MODB,RUN/ST:0XFF,AB, \LIB\START;
```

The final example sets the stack size to 24 (030 octal) bytes:

```
LINK STARTUP+FILE/ST:030, , ;
```

## 6.2.9 Setting the Maximum Allocation Space

**Format:**            `/cparmaxalloc:number`

**Remarks:**

The `/cparmaxalloc` option sets the maximum number of 16-byte paragraphs needed by a program when it is loaded into memory. The operating system uses this number when allocating space for a program prior to loading it.

LINK normally sets the maximum number of paragraphs to 65535. Since this represents all addressable memory, the operating system always denies the default settings and allocates no more space than is given by this option. This means any additional space in memory is free for other programs.

The *number* can be any number between 1 and 65535. It must be a decimal, octal, or hexadecimal number. Octal numbers must begin with a zero, and hexadecimal values must begin with a leading zero followed by a lowercase x. For example, 0x2B.

If *number* is less than the minimum number of paragraphs needed by the program, LINK ignores your request and sets the maximum value equal to the minimum needed. The minimum number of paragraphs needed by a program is never less than the number of paragraphs of code and data in the program.

**Abbreviation:** /c

**Example:**

The first sample sets the maximum allocation to 15 paragraphs:

```
LINK FILE/C:15,,;
```

The second example sets the maximum allocation to 255 (0FFh) paragraphs:

```
LINK MODA+MODB,RUN/CPARMAXALLOC:0XFF,AB;
```

### 6.2.10 Setting a High Start Address

**Format:**            /high

**Remarks:**

The /high option sets a program's starting address to the highest possible address in free memory. If you don't use the /high option, LINK sets the program's starting address as low as possible in memory.

**Abbreviation:** /h

**Example:**

This example sets the starting address of the program in *file.exe* to the highest possible address in free memory:

```
LINK STARTUP+FILE/HIGH, , ;
```

### 6.2.11 Allocating a Data Group

**Format:**            /dsallocate

**Remarks:**

The /dsallocate option directs the linker to reverse its normal processing when assigning addresses to items belonging to the group named DGROUP. Normally, LINK assigns the offset 0000h to the lowest byte in a group. If you use /dsallocate, LINK assigns the offset FFFFh to the highest byte in the group. The result is data that appear to be loaded as high as possible in the memory segment containing DGROUP.

Typically, you use /dsallocate option with the /high option to take advantage of unused memory before the start of the program. The linker assumes that all free bytes in DGROUP occupy the memory preceding the program. To use the group, you must set a segment register to the start address of DGROUP.

**Abbreviation:** /d

**Example:**

The following example directs the linker to place the program as high in memory as possible, then adjust the offsets of all data items in DGROUP so that they are loaded as high as possible within the group:

```
LINK STARTUP+FILE/HIGH/DSALLOCATE, , , EM+M  
LIBFP
```

### 6.2.12 Removing Groups from a Program

**Format:**           /nogroupassociation

**Remarks:**

The /nogroupassociation option directs LINK to ignore group associations when assigning addresses to data or code items.

**Abbreviation:** /nog

### 6.2.13 Setting the Overlay Interrupt

**Format:**            */overlayinterrupt:number*

**Remarks:**

The */overlayinterrupt* option sets the interrupt number of the overlay loading routine to *number*. This option overrides the normal overlay interrupt number (03Fh).

*Number* can be any integer value in the range from 0 to 255. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading \ero, and hexadecimal numbers must start with a leading zero followed by a lowercase x. For example, 0x3B.

MASM does not have an overlay manager. You can use this option only if you are linking with a run-time module from a language compiler that supports overlays. Check your compiler documentation, since you may not be able to use this option with some compilers.

**Abbreviation:**   */o*

**Example:**

The first example sets the overlay interrupt number to 255:

```
LINK FILE.0:225, , ,87+SLIBFP
```

The next example sets the overlay interrupt to 255 (0FFh):

```
LINK MODA+MODB,  
RUN/OVERLAY:0XFF,AB.MAP,EM+MLIBFP
```

## 6.2.14 Setting the Maximum Number of Segments

**Format:**            */segments:number*

**Remarks:**

The */segments* option directs the linker to process no more than *number* segments per program. If it encounters more than the given limit, the linker displays an error message, and stops linking. You use this option to override the default limit of 128 segments.

If you do not use */segments*, the linker allocates enough memory space to process up to 128 segments. If your program has more than 128 segments, you will need to set the segment limit higher to increase the number of segments that LINK can process. If you get the following LINK error message, you should set the segment limit lower:

Segment limit set too high.

The *number* can be any number from 1 to 1024. It must be a decimal, octal, or hexadecimal number. Octal numbers must have a leading zero, and hexadecimal numbers must start with a leading zero followed by a lower case x (0x4B).

**Example:**

This example sets the segment limit to 192:

LINKFILE/SE:192, , ;



### 6.2.15 Using DOS Segment Order

**Format:**            /dosseg

**Remarks:**

The /dosseg option causes LINK to arrange all segments in the executable file according to the DOS segment-ordering convention.

- All segments having the class name CODE are packed at the beginning of the executable file.
- Any other segments that do not belong to the group, DGROUP, are placed immediately after the CODE segments.
- All segments belonging to DGROUP are placed at the end of the file.

**Example:**

The following command causes the linker to create an executable file named *file.exe*, whose segments are arranged according to the DOS segment-ordering convention. The segments in the object files *start.obj*, and *test.obj*, and any segments copied from the libraries *math.lib* and *common.lib*, are arranged according to the same segment-ordering conventions.

```
LINK  
START+TEST/DOSSEG, , MATH+COMMON
```

## 6.3 How LINK Works

LINK creates an executable file by linking a program's code and data segments according to the instructions in the original source files. The linked segments form an *executable image* that is copied directly into memory when you run the program. The order and manner in which the linker copies segments to the executable file defines the order and manner in which it loads the segments into memory.

You can tell the linker how to link a program's segments by using a `SEGMENT` directive to supply segment attributes, or by using the `GROUP` directive to form segment groups. The directives define group associations, classes, and align and combine types that define the order and relative starting addresses of all segments in a program. This information works in addition to any information you supply through command line options.

The following sections explain the process that LINK uses to link segments and resolve references to items in memory.

### 6.3.1 Alignment of Segments

LINK uses a segment's align type to set the starting address for the segment. The align types are *byte*, *word*, *para*, and *page*. These types correspond to starting address at byte, word, paragraph, and page boundaries, representing addresses that are multiples of 1, 2, and 256, respectively. The default align type is *para*.

When the linker encounters a segment, it checks the align type before copying the segment to the executable file. If the align type is *word*, *para*, or *page*, the linker checks the executable image to see if the last byte copied ends at an appropriate boundary. If it doesn't, LINK pads the image with extra null bytes.

### 6.3.2 Frame Number

The linker computes a starting address for each segment in a program. The starting address is based on a segment's align type and on the size of the segments already copied to the executable file. The address consists of an offset and a canonical frame number, which specifies the address of the first paragraph in memory that contains one or more bytes of the segment.

A frame number is always a multiple of 16 (a paragraph address), and the offset is the number of bytes from the start of the paragraph to the first byte in the segment. For *byte* and *word align* types, the offset may be nonzero, but the offset is always zero for *para* and *page align* types.

The frame number of a segment can be obtained from a LINK file. The frame number is the first five hexadecimal digits of the start address specified for the segment.

### 6.3.3 Order Segments

LINK copies segments to the executable file in the same order that it encounters them in the object files. The linker maintains this order throughout the program unless it encounters two or more segments with the same class name. Segments with identical class names belong to the same class type, and are copied to the executable file as contiguous blocks.

### 6.3.4 Combined Segments

LINK uses combine types to determine whether two or more segments sharing the same name should be combined into a single large segment. The combine types are *public*, *stack*, *common*, *memory*, *at*, and *private*.

If a segment has a *public* combine type, the linker automatically combines it with any other segments that have the same name and belong to the same class. When LINK combines segments, it ensures that the segments are contiguous and that all addresses in the segments can be accessed using an offset from the same frame address. The result is same as if the segment were defined as a whole in the source file.

The linker preserves each segment's type, the linker carries out the same combine operation as for public segments. The only difference is that stack segments cause LINK to copy an initial stack-pointer value to the executable file. This stack-pointer value is the offset to the end of the first stack segment (or combined stack segment) the linker encounters.

If you use the *stack* type for stack segments, you do not need to give instructions to load the segment into the SS register.

If a segment has a *common* combine type, the linker combines it automatically with any other segments that have the same name and belong to the same class. When LINK combines common segments, however, it places the start of each segment at the same address, creating a series of overlapping segments. The result is a single segment no larger than the largest of the combined segments.

The linker treats segment with *memory* combine type exactly like segments with *public* combine types. The Macro Assembler (MASM), provides combine type memory for compatibility with linkers that support a separate combine type for memory segments.

A segment has a *private* combine type only if no explicit combine type is defined for it in the source file. LINK does not combine private segments.

### 6.3.5 Groups

Groups permit noncontiguous segments that do not belong to the same class to be addressable relative to the same frame address. When LINK encounters a group, it adjusts all memory references to items in the group so that they are relative to the same frame address.

Segments in a group do not have to be contiguous and do not have to belong to the same class. Nor do they have to have the same combine type. The only requirement is that all segments in the group fit within 64 KB.

Groups do not affect the order in which the segments are loaded. Unless you use class names and enter object files in the right order, there is no guarantee that the segments will be contiguous. In fact, the linker may place segments that do not belong to the group in the same 64 KB of memory. Although LINK does not explicitly check whether all segments in a group fit within this 64 KB of memory, the linker is likely to encounter a *fixup-overflow* error if this requirement is not met.

### 6.3.6 Fixups

Once the starting address of each segment in a program is known, and all segments combinations and groups have been established, the linker can fix up any unresolved references to labels and variables. To fix up unresolved references, the linker computes an appropriate offset and segment address and replaces the temporary values, generated by the assembler, with the new values.

LINK carries out fixups for four different references. These include *short*, *near self-relative*, *near segment-relative*, and *long*.

The size of the value to be computed depends on the type of reference. If LINK discovers an error in the anticipated size of a reference, it displays a *fixup-overflow* message. This error can happen for example, if a program attempt, by using a 16-bit offset, to reach an instruction in a segment that has a different frame address. The error can also occur if the segments in a group do not fit with a single 64 block of memory.

A short reference occurs in JMP Instructions that attempt to pass control to labeled instructions in the same segment or group. The target instructions must be no more than 128 bytes from the point of reference. The linker computes a signed, 8-bit number for this reference and displays an error message if the target instruction belongs to a different segment or group, or if the target is more than 128 bytes distant in either direction.

The near self-relative reference occurs in instruction that accesses data relative to the same segment or group. The linker computes a 16-bit offset for this reference and displays an error message if the data are not in the same segment or group.

A near segment-relative reference occurs in instruction that attempt to access data in a specified segment or group, or that is relative to a specified register. LINK computes a 16-bit offset for this reference and displays an error message if either of the following conditions exists: the offset of the target within the specified frame is greater than 64 KB or less than 0, or the beginning of the canonical frame of the target is not addressable.

A long reference occurs in CALL instructions that attempt to access an instruction in another segment or group. LINK computes a 16-bit frame address and 16-bit offset for this reference and displays an error message if either of the following conditions exists: the computed offset is greater than 64 KB or less than 0, or the beginning of the canonical frame of the target is not addressable.

LINK

## NOTES

DEBUG





The DEBUG utility is a debugging program that provides a controlled testing environment for binary and executable object files. Note that EDLIN, the disk operating system (DOS) line editor, is used to alter source files. DEBUG is EDLIN's counterpart for binary files.

DEBUG eliminates the need to reassemble a program or see if a problem has been fixed by a minor change. It allows you to alter the contents of a file or the contents of a central processing unit (CPU) register, and immediately re-execute a program to check the validity of the changes.

All DEBUG commands may be aborted at any time by pressing <Ctrl>-<C>. The <Ctrl>-<S> sequence suspends the display, so you can read it before the input scrolls away. Pressing any other key restarts the display. All these commands are consistent with the control character functions available at the DOS level.

## 7.1 How to Start DEBUG

DEBUG may be started two ways. By the first method (DEBUG), you type all commands in response to the DEBUG prompt (a hyphen). By the second method (Command Line), you type all commands on the line used to start DEBUG.

### 7.1.1 DEBUG

To start DEBUG, type:

```
DEBUG
```

DEBUG responds at the hyphen (-) prompt, signaling that it is ready to accept your commands. Since you didn't specify a filename, you can use other commands to work on current memory, disk sectors, or disk files.

When DEBUG is started, it sets up a program header at offset 0 in the program work areas. In previous versions of DEBUG, you could overwrite this header. You can still overwrite this header if you don't give a *filename* to DEBUG. If you are debugging a .com or .exe file, however, do not tamper with the program header below address 5, or DEBUG will terminate.

Do not restart a program after the following message is displayed:

```
Program terminated normally
```

You must reload the program with the N (name) and L (load) commands for it to run properly.

### 7.1.2 Command Line

To start DEBUG using a command line you must use the following syntax:

```
DEBUG[filename[arglist]]
```

For example, if you specify a *filename*, the following would be a typical command to start DEBUG:

```
DEBUG FILE.EXE
```

DEBUG would then load *file.exe* into memory starting at 0100h in the lowest available segment. The BX:CX registers are loaded with the number of bytes placed into memory.

If you do include a *filename*, you might also specify an *arglist*. An *arglist* is a list of filename parameters and switches that are passed to the program *filename*. When *filename* is loaded into memory, it is loaded as if it has been started with a command of the form, `DEBUG filename arglist`.

Here, *filename* is the file to be debugged, and *arglist* is the rest of the command line used when DEBUG calls and loads *filename* into memory.

## 7.2 Debug Commands and Parameters

Each DEBUG command consists of a single letter followed by one or more parameters. If a syntax error occurs in a DEBUG command, DEBUG reprints the command line and indicates the error with a caret (^) and the word "Error" as in the following example:

```
dsc:100 cs:110
^ Error
```

Note that when typing commands and parameters you may use any combination of uppercase and lowercase letters.

The DEBUG commands are listed in Table 7-1. Following this list, the commands and their parameters are described in greater detail.

**Table 7-1. DEBUG Commands.**

DEBUG Command	Function
A [address]	Assemble
C range address	Compare
D [range]	Dump
E address [list]	Enter
F range list	Fill
G [= address[address...]]	Go
H value value	Hex
I value	Input

Table 7-1. Continued.

L[address[drive:record record]]	Load
M range address	Move
N filename [filename]	Name
O value byte	Output
Q	Quit
R [register-name]	Register
S range list	Search
T [= address][value]	Trace
U [range]	Unassemble
W [address[drive:record record]]	Write

All DEBUG commands accept parameters, except the Q (quit) command. Parameters may be separated by delimiters (spaces or commas), but a delimiter is required only between two consecutive hexadecimal values. Thus, the following commands are equivalent:

```
DCS:100 110
D CS:100 110
D,CS:100,110
```

Table 7-2 shows the parameter names and the definition of each parameter.

**Table 7-2. DEBUG Parameters.**

Parameter	Definition
<i>drive:</i>	A one-digit hexadecimal value that indicates which drive a file will be loaded from or written to. The valid values are 0 - 3, where 0 = A:: 1 = B:: 2 = C:: 3 = D:.
<i>byte</i>	A two-digit hexadecimal value placed in or read from an address or register.
<i>record</i>	A one- to three--digit hexadecimal value that indicates the logical record number on the disk and the number of disk sectors to be written or loaded. Logical records correspond to sectors; however, since they represent the entire disk space, the numbering differs.
<i>value</i>	A hexadecimal value of up to four digits that specifies a port number or the number of times a command should repeat its functions.

Table 7-2. Continued.

<i>address</i>	A two-part designation containing either an alphabetic segment register or a four-digit segment address plus an offset value. You may omit the segment address, in which case the default segment DS is used for all commands except G, L, T, U, and W, for which the default segment is CS. All numeric values are hexadecimal.
<i>range</i>	Contains two addresses: for example <i>address address</i> ; or one <i>address</i> , an L and a <i>value</i> . The second type of <i>range</i> cannot be used if another hexadecimal value follows, since the hexadecimal value would be interpreted as the second <i>address</i> of the <i>range</i> .
<i>list</i>	A series of <i>byte</i> values or <i>strings</i> . <i>List</i> must be the last parameter on the command line.
<i>string</i>	Any number of characters enclosed in quotation marks. The quotation may be either single (') or double ("). If the delimiter marks must appear within a <i>string</i> , you must use the double quotation marks.



## ASSEMBLE

---

**Purpose:** Assembles 8086/8087/8088 mnemonics directly into memory.

**Format:** A [address]

**Remarks:**

If a syntax error is found, DEBUG responds with the following message, then redisplay the current assembly address:

^ Error

All numeric values are hexadecimal and you must type them as 1 - 4 characters. Also, you must specify prefix mnemonics *in front of* the opcode to which they refer. You may type them on a separate line, however.

The segment override mnemonics are CS:, DS:, ES:, and SS:. The mnemonic for the far return is RETF. String manipulation mnemonics must explicitly state the string size. For example, use MOVSW to move word strings, and MOVSB to move byte strings.

The assembler will automatically assemble short, near or far jumps and calls, depending on byte displacement, to the destination address. You may override these jumps and calls by using a NEAR or FAR prefix as in the following example:

```
0100:0500 JMP 502      ; a 2-byte short jump
0100:0502 JMP NEAR 505 ; a 3-byte near jump
010:505   JMP FAR 50A  ; a 5-byte far jump
```

You may abbreviate the NEAR prefix to NE, but the FAR prefix cannot be abbreviated.

DEBUG cannot tell whether some operands refer to a word memory location or to a byte memory location. In this case, the data type must be explicitly stated with the prefix, WORD PTR or BYTE PTR. Acceptable abbreviations are WO and BY. For example:

```
NEG  BYTE PTR [128]
DEC  WO [SI]
```

DEBUG also cannot tell whether an operand refers to a memory location or to an immediate operand. So, it uses the common convention that operands enclosed in square brackets refer to memory. For example:

```
MOV  AX,21      ; Load AX with 21h
MOV  AX, [21]   ; Load AX with the contents
                ; of memory location 21h
```

Two popular pseudo-instructions are available with the assemble command: the DB opcode, which assembles byte values directly into memory, and the DW opcode, which assembles word values directly into memory. Following are examples of both:

```
DB  1,2,3,4,"THIS IS AN EXAMPLE"
DS  'THIS IS A QUOTATION MARK:'
DB  "THIS IS A QUOTATION MARK:"

DW  1000,2000,3000,"BACH"
```

The a command supports all forms of register indirect commands. For example:

```
ADD  BX,34[BP + 2].[SI-1]
POP  [BP + DI]
PUSH [SI]
```

All opcode synonyms are also supported, as in the next example:

```
LOOPZ 100
LOOPE 100

JA     200
JNBE   200
```

For 8087 opcodes, the WAIT or FWAIT prefixes must be explicitly specified, as in these examples:

```
FWAIT FADD ST,ST(3) ; This line assembles
                    ; an FWAIT prefix.
LD TBYTE PTR [BX]   ; This line does not.
```

## COMPARE

---

**Purpose:** Compares the portion of memory specified by *range* to a portion of the same size beginning at the specified *address*.

**Format:** Cn *range address*

**Remarks:**

If the two areas of memory are identical, there is no display, and DEBUG returns with the DOS prompt. If there *are* differences, they are displayed in this format

*address1 byte1 byte2 address2*

**Example:**

The following commands have the same effect:

C100,1FF 300

or

C100L100 300

Each command compares the block of memory from 0100 to 01FFh with the block of memory of 0300 to 03FFh.

## DUMP

---

**Purpose:** Displays the contents of the specified *range* of memory.

**Format:** D[range]

**Remarks:**

If you specify a *range* of addresses with the dump command, the contents of the *range* are displayed. If you don't use parameters with the D command, 128 bytes are displayed at the first address (DS:100) after the address displayed by the previous D command.

The dump is displayed in two portions: a hexadecimal dump (each byte is shown in hexadecimal value) and an ASCII dump (the bytes are shown in ASCII characters). Nonprinting characters are denoted by a period (.) in the ASCII portion of the display. Each display line shows 16 bytes, with a hyphen between the eighth and ninth bytes. At times, displays are split in this chapter to fit them on a page. Each displayed line begins on a 16-byte boundary.

**Example:**

If you type the command:

```
DS:100 110
```

DEBUG displays the dump in the following format:

```
04BA:0100 42 45 52 54 41 ... 4E 44 JAMES BOND
```

If you type the D command, the display is formatted as described above. Each line of the display begins with an address, incremented by 16 from the address on the previous line.

Each subsequent D (typed without parameters) displays the bytes immediately following those last displayed.

If you type the following command, the display is formatted as described above, but 020h bytes are displayed:

```
DCS:100 L 20
```

If you type the following command, the display is formatted as described above, but all the bytes in the range of lines from 0100h to 0115h in the CS segment are displayed:

```
DCS:100 115
```

## ENTER

---

**Purpose:** Enters byte values into memory at the specified address.

**Format:** Eaddress[list]

**Remarks:**

If, when using the enter command, you type the option *list* of values, the byte values are replaced automatically. If an error occurs, no byte values are changed.

If you type the *address* without the optional *list*, DEBUG displays the *address* and its contents, repeats the *address* on the next line, and waits for your input. At this point, the E command waits for you to perform one of the following actions

- Replace a byte value with a value you type. Simply type the value after the current value. If the one you type is not a legal hexadecimal value or if it contains more than two digits, the illegal or extra character is not echoed.
- Press < Space > to advance the next byte. To change the value, simply type the new value as described in the previous action. If, when you press < Space >, you move beyond an 8-byte boundary, DEBUG starts a new display line with the address displayed at the beginning.
- Type a hyphen (-) to return to the preceding byte. If you decide to change a byte behind the current position, typing the hyphen returns the current position to the previous byte. When you type the hyphen, a new line is started with its address and byte value displayed.
- Press < Enter > to terminate the E command. < Enter > may be pressed at any byte position.

**Example:**

Suppose you type the following command:

```
ESC; 100
```

Now suppose that DEBUG displays the following:

```
04BA:0100 EB._
```

To change this value to 41, type the number 41 at the cursor as shown:

```
04BA:0100 EB.41_
```

To step through the subsequent bytes, you would press <Space> until you saw the following:

```
04BA:0100 EB.41 10. 00. BC._
```

To change BC to the number 42, for instance, you would type the number at the cursor, as follows:

```
04BA:0100 EB.41 10. 00. BC.42_
```

Notice that value 10 should be 06Fh. To correct this value, you would type the hyphen as many times as needed to return to byte 0101h (value 10), then replace 10 with 06F:h

```
04BA:0100 EB.41 10. 00. BC.42-
04BA:0102 00.-
04BA:0101 10.6F
```

Pressing <Enter> ends the end command and returns you to the DEBUG command level.



## FILL

---

**Purpose:** Fills the addresses in the specified *range* with the values in the specified *list*.

**Format:** Frange list

**Remarks:**

If the *range* contains more bytes than the number of values in the *list*, the *list* will be used repeatedly until all bytes in the *range* are filled.

If the *list* contains more values than the number of bytes in the *range*, the extra values in the *list* are ignored. If any of the memory in the *range* is not valid (bad or nonexistent), the error will occur in all succeeding locations.

**Example:**

04BA:0100 L 100 42 45 52 54 41

DEBUG would now fill memory locations 04BA:100 through 04BA:1FF with the bytes specified. The file values would then be repeated until the 0100h bytes were filled.

## GO

---

**Purpose:** Executes the program currently in memory.

**Format:** G[=address[address]]

**Remarks:**

If you type the go command by itself, the program currently in memory executes as if it had run outside DEBUG.

If you set =*address*, execution of the G command begins at the address specified. The equal sign (=) is required so DEBUG can distinguish the start address from the breakpoint address.

With the other option *addresses* set, execution stops at the first *address* encountered, regardless of that address's position in the list of addresses that halt execution or program branching. When program execution reaches a breakpoint, the registers, flags, and decoded instructions are displayed for the last instructions executed. The result is the same as if you had typed the R command.

You may set up to ten breakpoints, but only addresses containing the first byte of an 8086 opcode. If you set more than ten breakpoints, DEBUG returns the BP error message.

The user stack pointer must be valid and must have six bytes available for this command. The G command uses an IRET instruction to cause a jump to the program under test. The user stack pointer is set, and the user flag, Code Segment register, and Instruction Pointer are pushed on the user stack. (If the user stack is not valid or is too small, the operating system may crash.) An interrupt code (0CCh) is placed at the specified breakpoint address(es).

When DEBUG encounters an instruction with the breakpoint code, it restores all breakpoint addresses to their original instructions. If you don't halt execution at one of the breakpoints, the interrupt codes are not replaced with the original instructions.

**Example:**

Suppose you type the following command:

GCS:7550

The program currently in memory would execute up to the address 7550 in the CS segment. DEBUG would then display registers and flags, after which the G command would terminate.

After DEBUG has encountered a breakpoint, if you type G command again, the program runs again as if you typed the filename at the DOS prompt. The only difference is that program execution begins at the instruction after the breakpoint, rather than at the usual start address.

## HEX

---

**Purpose:** Performs hexadecimal arithmetic on the two specified parameters.

**Format:** Hvalue value

**Remarks:**

First, DEBUG add the two parameters, then subtracts the second parameter from the first. The results of these actions are displayed on one line -- first the sum then the difference.

**Example:**

Suppose you type the following command:

H19F 10A

In response, DEBUG would perform the calculations and display the following results:

02A9 0095

## INPUT

---

**Purpose:** Inputs and displays one byte from the port specified by *value*.

**Format:** I *value*

**Remarks:**

One 16-bit port address is allowed.

**Example:**

Suppose you type the following command:

I2F8

Suppose also that the byte at the port is 42h. DEBUG would input the byte and then display the following.

42

## LOAD

---

**Purpose:** Loads a file into memory.

**Format:** L[address[drive:record record]]

**Remarks:**

Set BX:CX to the number of bytes read. The file must have been named either when you started DEBUG or with the name (N) command. Both the DEBUG invocation and the N command format a filename properly in the normal format of a File Control Block at CS:5Ch.

If you use the L command without any parameters, DEBUG loads the file into memory beginning at address CS:100 and sets BX:CX to the number of bytes loaded. If you type L with an address parameter, loading begins at the memory location specified by the *address*. If you use the L command with all parameters included, absolute disk sectors are loaded, instead of a file.

Each *record* is taken from the specified *drive*: (the drive designation is numeric: 0 = A:, 1 = B:, 2 = C:, and 3 = C:)/DEBUG begins loading with the first specified *record*, and continues until the number of sectors in the second *record* have been loaded.

**Example:**

Suppose once you have started DEBUG that you type the following commands:

```
-NFILE.COM
```

Now to load *file.com*, you would simply type the L command.

DEBUG would then load the file and display the DEBUG prompt. Suppose now that you want to load only portions of a file or certain records from a disk. To do this, you would type:

```
L04BA:100 2 0F 6D
```

DEBUG would then load 109 (06Dh) records, beginning with logical record number 15, into memory beginning at address 04BA:0100. Then once it had loaded the records, DEBUG would simply return the hyphen (-) prompt.

If the file has an .exe extension, it would be relocated to the load address specified in the header of the .exe file. The *address* parameter is always ignored for .exe file. The header itself is stripped of the .exe file before it is loaded into memory. So, the size of an .exe file on disk will differ from its size in memory.

If the file was named by the N (name) command, or specified when you started DEBUG, as a .hex file, then typing L with no parameters would cause DEBUG to load the file beginning at the address specified in the .hex file. If the L command included the option *address*, DEBUG would add the address specified in the L command to the address found in the .hex file to determine the start address at which to load the file.

## MOVE

---

**Purpose:** Moves a block of memory specified by *range* to the location beginning at the specified *address*.

**Format:** Mrange address

**Remarks:**

Overlapping moves (moves where part of the block overlaps some of the current addresses) are always performed without loss of data. Addresses that could be overwritten are removed first. For moves from higher to lower addresses, the sequence of events is to move the data beginning at the block's lowest address and then work toward the highest. For moves from lower to higher addresses, the sequence is to first move the data beginning at the block's highest address and then working toward the lowest.

Note that if the addresses in the block being moved will not have new data written to them, the data in the block *before the move* will remain. The M (move) command copies the data from one area to another, in the sequence described, and writes over the new addresses. This action is why the sequence of the move is important.

**Example:**

Suppose you type the following command:

```
MCS:100 110 CS:500
```

In response, DEBUG would first move address CS:110 to CS:510, then move CS:10F to CS50F, and so on until CS:100 is move to CS:500. To review the results of the move, you could type the D command, using the same address as you used with the M command.



## NAME

---

**Purpose:** Sets file names.

**Format:** Nfilename [filename...]

**Remarks:**

The name command performs two functions. First, it assigns a filename or a later load or write command. If you start DEBUG without naming a file to be debugged, you must type the command Nfilename before a file can be loaded. Second, the N command assigns filename parameters to the file being debugged. In this case, N accepts a list of parameters used by the file being debugged.

Note that these two functions overlap. Consider, for example, the following set of DEBUG commands:

```
-NFILE1.EXE  
-L  
-G
```

The N command would use these commands to perform the following steps:

- It would first assign the filename *file1.exe* to the file to be used in any later L or W commands.
- It would also assign the *file2.exe* filename to the first filename parameter used by any program that is later debugged.
- The L command would then cause *file1.exe* to run as if *file1.exe* had been typed at the command level.

A more useful chain of commands might look like this:

```
-N FILE1.EXE  
-L  
-N FILE2.DAT FILE3.DAT  
-G
```

In this example, the N command sets *file1.exe* as the file name for the subsequent L command, which loads *file1.exe* into memory. The N command is then used again, this time to specify the parameters to be used by *file1.exe*. Finally, when the G command is run, *file1.exe* is executed as if *file1 file2.dat file2.dat* had been typed at the DOS command level.

Note that if you were to execute a W command now, then *file1.exe* -- the file being debugged -- would be saved with the name *file2.dat*. To avoid this kind of result, you should always execute an N command before either an L or W command.

There are four regions of memory that can be affected by the N command:

CS:5C	FCB for file 1.
CS:6C	FCB for file 2.
CS:80	Count of characters.
CS:81	All characters typed.

The first filename parameter that you specify for the N command has a file control block (FCB) set up at CS:5C. If you name a second file name parameter, an FCB is set up for this parameter beginning at CS:6C. The number of characters typed in the N command (exclusive of the first character, N) is given a location at CS:80.

The actual stream of characters given by the N command begins at CS:81. Note that this stream of characters may contain switches and delimiters that would be legal in any command typed at the DOS prompt.

**Example:**

A typical use of the N command is as follows:

```
-DEBUG PROG.COM  
-NPARAM1 PARAM2/C  
-G  
-
```

In this case, the G command executes the file in memory as if you had typed the following command line:

```
PROG PARAM1 PARAM2/C
```

Testing and debugging there reflect a normal run-time environment for *prog.com*.

## OUTPUT

---

**Purpose:** Send the specified *byte* to the output port specified by *value*.

**Format:** 0value byte

**Remarks:**

A 16-bit port address is allowed.

**Example:**

Suppose you want to DEBUG to output the byte value 4F to output port 2F8. To do this you could simply type the following command:

```
02F8 4F
```

## QUIT

---

**Purpose:** Terminates the DEBUG utility.

**Format:** Q

**Remarks:**

The Q (quit) command takes no parameters and exits DEBUG without saving the file you are currently working with. You return to the DOS prompt.

**Example:**

To end the debugging session, type:

Q <Enter>

## REGISTER

---

**Purpose:** Displays the contents of one or more CPU registers.

**Format:** R[register-name]

**Remarks:**

If you do not type a *register-name*, the R (register) command dumps the register storage area and displays the contents of all registers and flags.

If you do type a *register-name*, the 16-bit value of that register is displayed in hexadecimal, and a colon then appears as a prompt. You can now either type a *value* to change the register, or press <Enter> if you don't want a change.

Following is a list of the valid *register-names*:

AX	BP	SS	
BX	SI	CS	
CX	DI	IT	(IP and PC both refer
DX	DS	PC	to the Instruction
SP	ES	F	Pointer.)

Any other entry for *register-name* results in a BR error message.

If you type F as the register-name, DEBUG displays each flag with a two-character alphabetic code. To change any flag, type the opposite two-letter code. The flags are then either set or cleared.

The flags are listed in Table 7-3 with their codes for SET and CLEAR:

**Table 7-3. DEBUG Flag Codes.**

Flag Name	Set	Clear
Overflow	OV	NV
Direction	DN Decrement	UP Increment
Interrupt	EI Enabled	DI disabled
Sign	NG Negative	PL Plus
Zero	ZR	NZ
Aux. Carry	AC	NA
Parity	PE Even	PO Odd
Carry	CY	NC

Whenever you type the RF command, the flags are displayed in a row at the beginning of a line. At the end of the list of flags, DEBUG displays a hyphen (-).

You may enter new flag values in any order as alphabetic pairs. You do not have to leave spaces between these values. To exit the R command, press <Enter>. Any flags you did not specify new values for remain unchanged.

If you type more than one value for a flag, DEBUG returns a DF error message. If you enter a flag code other than one of those shown in the table above, DEBUG returns a BF error message. In both cases, the flags up to the error in the list are changed; those flags at and after the error are not.

When you start DEBUG, the segment registers are set to the bottom of free memory, the Instruction Pointer is set to 0100H, all flags are cleared, and the remaining registers are set to zero.

### Example:

If you type the following command, DEBUG displays all registers, flags, and the decoded instruction for the current location:

R

If the location is CS:11A, for example, the display will look similar to this:

```
AX = 0E00 BX = 00FF CX = 0007 DX = 01FF SP = 039D BP = 0000
SI = 005C DI = 0000 DS = 04BA ES = 04BA SS = 04BA CS = 04BA
IP = 011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21      INT    21
```



If you type:

RF

DEBUG will display these flags:

NV UP DI NG NZ AC PE NC - \_

Now, you could type any valid flag designations, in any order, with or without spaces. For example:

NV UP DI NG NZ AC PE NC - PLEICY

In response, DEBUG would display the DEBUG prompt. To see the changes, type either the R or RF command:

RF

NV UP EI PL NZ AC PE CY - \_

Press <Enter> to leave the flags this way or to specify different flag values.

## SEARCH

---

**Purpose:** Searches the specified *range* for the specified *list* of bytes.

**Format:** Srange list

**Remarks:**

The *list* may contain one or more bytes, each separated by a space or comma. If the *list* contains more than one byte, only the first address of the byte string is returned. If the *list* contains only one byte, all addresses of the byte in the *range* are displayed.

**Example:**

Suppose you type the following command:

```
SCS:100 110 41
```

DEBUG would display a response similar to this:

```
04BA:0104  
04BA:0100D  
-TYPE:
```

## TRACE

---

**Purpose:** Executes one instruction and displays the contents of all registers, flags, and the decoded instruction.

**Format:** T[=address] [value]

**Remarks:**

If you include the =*address* option in the T (trace) command, tracing occurs at the specified =*address*. The *value* option causes DEBUG to execute and trace the number of steps specified by *value*.

The T command uses the hardware trace mode of the 8086 or 8088 microprocessor. Consequently, you may also trace instructions stored in read-only memory (ROM).

**Example:**

Suppose you type the following command:

T

In response, DEBUG would return a display of the registers, flags, and decoded instruction for that one instruction. Assuming for this example, that the current position is 04BA:011A, DEBUG might return the following display:

```
AX=0E00 BX=00FF CX=0077 DX=01FF SP=039D BP=0000
SI=005C DI=0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A  NV UP DI NG NZ AC PE NC
04BA:011A  CD21      INT    21
```

If you type the following command, DEBUG executes system (010h) instructions beginning at 001Ah in the current segment, and then displays all registers and flags for each instruction as it is executed. The display scrolls away until the last instruction is executed and then stops. Now you can see the register and flag values for the last few instructions performed:

```
T=001A 10
```

Remember that if you want to study the registers and flags for any instruction, at any time, press <Ctrl>-<S> to stop display from scrolling.

## UNASSEMBLE

---

**Purpose:** Disassembles bytes and displays the source statements that correspond to them, with addresses and byte values.

**Format:** U[range]

**Remarks:**

The display of disassembled code looks like a listing for an assembled file. If you type the U (unassemble) command without parameters, 20 hexadecimal bytes are disassembled at the first address after that displayed by the previous U command. If you type the U command including the *range* parameter, then DEBUG disassembles all bytes in *range*. But if you specify *range* only as an address, then 020h bytes are disassembled.

**Example:**

Suppose you type the following command:

```
U04BA:100 L10
```

In response, DEBUG would disassemble 16 bytes, beginning at address 04BA:0100:

```
04BA:0100 206472 AND [SI + 72],AH
04BA:0103 69      DB 69
04BA:0104 7665    JBE 016B
04BA:0106 207370 AND [BP + DI + 70],DH
04BA:0109 65      DB 65
04BA:010A 63      DB 63
04BA:010B 69      DB 69
04BA:010C 69      DB 69
04BA:010D 69      DB 69
04BA:010E 63      DB 63
04BA:010F 61      DB 61
```

Now suppose you type the following:

```
U04BA:0100 010B
```

The display would now show:

```
04BA:0100 206472 AND [SI + 72],AH
04BA:0103 69      DB 69
04BA:0104 7665    JBE 016B
04BA:0106 207370 AND [BP + DI + 70],DH
```

If the bytes in some addresses are altered, the disassembler alters the instruction statements. You can then type the U command for the changed locations, the new instructions viewed, and the disassembled code used to edit the source file.

## WRITE

---

**Purpose:** Writes the file being debugged to a disk file.

**Format:** W[address[drive: record record]]

**Remarks:**

If you do not use parameters with the W (write) command, BX:CX must already be set to the number of bytes to be written; The file is written beginning from CS:100. If you type the W command with just an *address*, then the file is written beginning at that *address*. If you have used a G or T command, you must reset BX:CX before using the W command without parameters.

Note that If a file is loaded and modified, the name, length, and starting address are all set correctly to save the modified file (as long as the length has not changed).

You must have named the file either with the Initial DEBUG startup command or with the N command. Both DEBUG startup command and the N command properly format a filename in the normal format of a File Control Block at CS:5C.

If you include parameters when you use the `W` command, the write begins from the memory address specified. the file is written to the specified drive. `DEBUG` writes the file beginning at the logical record number specified by the first *record*. `DEBUG` then continues to write the file until the number of sectors specified in the second *record* have been written.

### Example:

If you type the following command, `DEBUG` will write the file to disk and then display the `DEBUG` prompt:

```
W
```

Two examples are shown below:

```
W
```

```
-
```

```
WCS:100 1 37 2B
```

`DEBUG` writes out the contents of memory to the disk in drive B:, beginning with the address `CS:100`. The data written out starts in the disk logical record number `037h` and consists of `02Bh` records. When the write is complete, `DEBUG` displays the following prompt:

```
WCS:100 1 37 2B
```

```
-
```



## 7.3 DEBUG Error Messages

During a DEBUG session, you may receive any of the following error messages. Each error ends the DEBUG command under which it occurred, but does not end DEBUG itself.

### BF

Bad flag. You attempted to change a flag, but the characters you typed were not one of the acceptable pairs of flag values. See the R command for the list of acceptable flag entries.

### BP

Too many breakpoints. You specified more than ten breakpoints as parameters to the G command. Retype the G command using ten or few breakpoints.

### BR

Bad register. While using the R command, you typed an invalid register name.

### DF

Double flag. You typed two values for one flag. You may specify a flag value only once per RF command.

CONFIG.SYS



The configuration file, CONFIG.SYS, provides instructions for MS-DOS to use *your* system. The instructions, for example, tell MS-DOS you have an external, mini-floppy drive or tell MS-DOS the last valid drive letter you will be using is M:. CONFIG.SYS is run automatically when your PC is booted.

The following CONFIG.SYS file is provided on your utilities diskette:

```
DEVICE = REMM.SYS
DEVICE = REX.SYS 384
DEVICE = FASTDISK.SYS /M = 360 /EXTM
```

If you boot from floppies, CONFIG.SYS should be copied to your MS-DOS boot diskette. If you boot from a hard drive, CONFIG.SYS should be copied to your root directory.

Using EDLIN, you can modify CONFIG.SYS to contain additional instructions such as:

```
BUFFERS = 20
FILES = 10
DEVICE = REMM.SYS
DEVICE = REX.SYS 384
DEVICE = FASTDISK.SYS /M = 360 /EXTM
DEVICE = \BIN\NETWORK.SYS
BREAK = ON
LASTDRIVE = E
```

Each instruction in the file is a statement. Each statement is explained in the following pages.

When you have finished modifying CONFIG.SYS, you must reboot your system so MS-DOS will read the new instructions.

## BREAK

---

**Purpose:** Determines which activities will be stopped when < Control > - < C > is pressed.

**Format:** BREAK=[on]

or

BREAK=[off]

where *off* is the default.

**Remarks:**

When BREAK is set to OFF, you can press < Control > - < C > to stop an activity when MS-DOS is reading from the keyboard, writing to the screen, or writing to a printer.

When BREAK is set to ON, you can press < Control > - < C > to stop an activity when MS-DOS is reading from the keyboard, writing to the screen, writing to a printer, or reading from or writing to a disk.

With BREAK set to ON you could abort a program stuck in a loop by pressing < Ctrl > - < C >. With BREAK set to OFF, pressing < Ctrl > - < C > would not break the loop because MS-DOS is not reading the keyboard or writing to a screen or printer. Typically, if you will be programming, set BREAK to ON. If you will not be programming, set BREAK to OFF to prevent a disk read or write from being aborted by mistake.

## BUFFERS

---

**Purpose:** Set the number of disk buffers allocated in memory when you start the system.

**Format:** BUFFERS= *N*

where *n* is the number of buffers. The default number is 2. For applications such as word processors, a number between 10 and 20 will provide the best performance. If you plan to create many subdirectories, increase the buffer to a number between 20 and 30.

**Remarks:**

A disk buffer is a block of memory where MS-DOS holds data being read from or written to a disk. When an application program reads or writes data, MS-DOS checks the buffers for the data before accessing the disk. (Data would be in a buffer if it was previously read or written.) If the data is in the buffer, it can be transferred to the application area without accessing the disk. The transfer of data from the buffer instead of the disk is faster because no mechanical movements are required.

The most efficient number of buffers will depend on your application programs. Some programs read randomly and the availability of data in the buffer speeds processing. Other programs read sequentially and checking buffers that do not contain the desired data slows processing. Ranges for best performance are given in the format description. Check your application program's manual for a suggested buffer number.

BUFFERS increase the amount of MS-DOS resident in memory by 520 bytes per buffer.

## COUNTRY

---

**Purpose:** Controls time, date, currency, and case conversion.

**Format:** COUNTRY = XXX

where xxx is the country code. Valid country codes are as follows:

Code	Country
001	United States
031	Netherlands
032	Belgium
033	France
034	Spain
039	Italy
041	Switzerland
044	United Kingdom
045	Denmark
046	Sweden
047	Norway
049	Germany
061	Australia
358	Finland
972	Israel

The default code is 001.

**Example:**

To set the currency, time, date, and case conversion to French, type:

COUNTRY=033

When MS-DOS prompts for the date, it will be in the dd-mm-yy format.



## DEVICE

---

**Purpose:** Instructs MS-DOS to recognize additional capabilities and/or hardware devices in your system.

DEVICE statements consist of installable device drivers (unique to each device type) and their options. The standard, installable device drivers provided with your Premium/286 are:

REMM.SYS  
REX.SYS  
ANSI.SYS  
DRIVER.SYS  
VDISK.SYS

REMM.SYS and REX.SYS are provided on the utility software diskette and are explained in the *AST Premium/286 Utility Software User's Manual*. ANSI.SYS, DRIVER.SYS, and VDISK.SYS are provided on your MS-DOS diskette and are discussed on the following pages.

**Driver:** ANSI.SYS

ANSI.SYS allows your screen and keyboard to emulate American National Standards Institute (ANSI) devices. ANSI.SYS should be added to your CONFIG.SYS file if you plan to use terminal emulation software or if your application program requires it.

**Format:** DEVICE = [d:][path]ANSI.SYS

where:

*d:* is the drive on which ANSI.SYS resides.

*path:* is the directory/subdirectory containing ANSI.SYS.

**Remarks:**

The size of DOS in memory increases by the size of ANSI.SYS.

**Driver:** DRIVER.SYS

DRIVER.SYS has two uses. One, it instructs DOS to recognize external, mini-floppy drives. (An external, mini-floppy drive is a drive physically outside of the Premium/286 that uses 3-1/2 inch diskettes.) You will need to add DRIVER.SYS to your CONFIG.SYS file if you have an external, mini-floppy drive. Two, it instructs DOS to assign logical drive letters to either internal or external floppy drives. For example, with DRIVER.SYS you can create two logical drives on a single floppy drive. Then, you can use the same floppy drive to transfer data from one diskette to another.

**Format:** DEVICE=DRIVER.SYS /D:<dd> [/C]  
 [/F:<ff>] [/H:<hh>] [/S:<sss>]  
 [/T<ttt>]

where *dd* is the physical, floppy drive number. Drives are numbered according to the following conventions:

- 0 First physical floppy drive (always internal). MS-DOS references it as drive A:
- 1 Second physical floppy drive (either internal or external).
- 2 Third physical floppy drive (always external).
- 3-25 MS-DOS can recognize up to 26 drives (one for each letter in the alphabet). However, the maximum valid number (25) will decrease by the number of DOS partitions on your fixed disk (if any).

**Switches:**

**/C** Requires changeline (door lock) support. Refer to your external floppy drive's manual to determine whether or not this switch should be included.

**/F** Device type.

where:

0 = 320/360 KB

1 = 1.2 MB

2 = 720 KB

3 = 1.44 MB

If you do not specify the /F switch, DRIVER.SYS uses a default of 720 KB.

**/H** Maximum head number. Refer to your floppy's manual for this value. Valid values are 1 to 99. The default value is 2.

**/S** Number of sectors per track. Refer to your floppy drive's manual for this value. Valid values are 1 to 99. The default value is 9.

**/T** Number of tracks per side. Refer to your floppy drive's manual for this value. Valid values are 1 to 999. The default value is 80.

**Example:**

You have two, internal floppy drives and you want to add a 720 KB, external mini-floppy drive. Type the following command in CONFIG.SYS:

```
DEVICE=DRIVER.SYS /D:2
```

You want to assign two, logical drive letters to your first, internal floppy drive. (You also have another floppy and a fixed disk in your system.) Add the following lines to your CONFIG.SYS file:

```
DEVICE=DRIVER.SYS /D:0  
DEVICE=DRIVER.SYS /D:0
```

By entering the command twice, MS-DOS will assign two, logical drive letters to the physical drive. This means you can copy data from a diskette in the floppy drive to another diskette in the same drive with the following command:

```
DISKCOPY A: D:
```

**Driver:** VDISK.SYS

VDISK.SYS allows portions of the computer's memory to store data. Each portion is referred to as a virtual disk. Virtual disks work like regular disk. Each virtual disk has an identify drive letter and a directory. However, you lose the contents of a virtual disk each time you boot the system or lose power to your computer.

**Format:** DEVICE=VDISK.SYS [bbb] [sss] [ddd]  
[/E[:m]]

where:

*bbb* is the size of the virtual disk in kilobytes (KB). The valid values are from 64 to the total amount of computer memory. The default value is 64.

If there is less than 64 KB of available memory, an error messages displays and the virtual disk is not installed. If you specify a size greater than the amount of memory in you computer, VDISK uses 64 KB. If you specify a size that leaves less than 64 KB of memory, VDISK adjusts the slze so 64 KB of memory remains.

*sss* is the sector size in bytes. Valid values are 128, 256, or 512. The default value is 128.

*ddd* is the number of directory entries (files) the virtual disk can hold. Valid values are 2 to 512. The default value is 64.

If *ddd* is not a multiple of the sector size (*sss*) then VDISK adjusts the number of directory entries upward to the nearest sector boundary.

If *ddd* is too small to hold the file allocation table (FAT), the directory and two additional sectors, VDISK decreases the number of directory entries until the conditions are met. If *ddd* is not large enough to hold at least one sector of directory entries, the FAT, and two additional sectors, VDISK issues an error message and does not install the virtual disk.

One directory entry holds a volume label.

*/E* instructs VDISK to use extended memory. Extended memory is memory at or above 1 MB. */E* causes VDISK to place the virtual disk buffer in extended memory and install the device driver in low memory (0-640 KB).

*m* is the maximum number of sectors of data transferred at a time. Valid values are 1 to 8. The default value is 8.

**Remarks:**

Virtual disks perform quickly because they operate at the speed of the computer's memory.

If the driver is installed in low memory (the /E switch is not used), each virtual disk increases the resident size of DOS by 720 bytes plus the size of the buffer you specify.

You can install more than one virtual disk in extended memory by including the appropriate number of DEVICE = VDISK.SYS/E commands in your CONFIG.SYS file. VDISK installs the first device drive at the 1 MB boundary, the second immediately following and so on. However, you must also allow for the virtual disk drive letter in your LASTDRIVE statement.

You do not need to format virtual disks, because DOS installs each disk in the formatted form.

To use a virtual disk, copy the files you need to the virtual disk. When you have finished your task, copy the files from the virtual disk to a floppy or hard disk. *Remember, when you reboot or turn the system off the data in virtual disk is lost.*

**Example:**

To install a 160 KB virtual disk with 256 byte sectors and 32 directory entries, type the following command:

```
DEVICE=VDISK.SYS 160 256 32
```



## FCBS

---

**Purpose:** Determines the maximum number of file control blocks (FCBs) opened files open at the same time.

**Format:** FCBS= x,y

where:

x is the maximum number of FCB-opened files that can be open at the same time. Valid values are from 1 to 255. The default value is 4.

y is the number of FCB-opened files that cannot be closed automatically if an application tries to FCB-open more than x. (The first y FCB-open files are protected from being closed.) Valid values are from 1 to 255. The default value is 0. y must be smaller than or equal to x.

**Remarks:**

This statement is necessary only if you are using MS-DOS version 2.1 or lower. (MS-DOS version 3.x is provided with your Premium/286.)

**Example:**

The following statement allows a maximum of four, FCB-opened files to be open at the same time. The first two, FCB-opened files cannot be closed if the program tries to open more than four.

FCB=4,2

## FILES

---

**Purpose:** Determines the maximum number of file handles open at the same time.

**Format:** FILES=x

where x is the number of open file handles the system can access. Valid values are from 8 to 255. The default value is 8. Values higher than 20 function only if the SHARE command is in use. (See Section 3.3.)

**Example:**

To set the FILES to 20, type:

FILES=20

## LASTDRIVE

---

**Purpose:** Set the last valid drive letter MS-DOS will recognize.

**Format:** LASTDRIVE=x

where x is any letter from A to Z. x must be greater than the actual number of drives (physical and logical) you plan to use. For example, if you plan to use four drives, x must be E or greater.

**Remarks:**

If you plan to use a network or virtual disks, a LASTDRIVE statement should be added to your CONFIG.SYS file. However, MS-DOS allocates a data structure for each drive specified, so do not specify more drives than necessary.

**Example:**

To set the last drive to M, type:

LASTDRIVE=M

## SHELL

---

**Purpose:** Use command processor specified in statement rather than standard, MS-DOS COMMAND.COM.

**Format:** SHELL=[d:] [path] filename [.ext]

**Remarks:**

The SHELL statement should be added to your CONFIG.SYS file if you write your own command processor (the MS-DOS command processor is COMMAND.COM). MS-DOS will start the processor specified in the path instead of reading the standard COMMAND.COM.

**Example:**

To use the command processor ANYSHELL.COM in the \BIN directory, type:

```
SHELL=\BIN\ANYSHELL.COM
```

## NOTES

**AST RESEARCH, INC.**

**Product Comment Form**

**AST Premium/286™  
MS-DOS User's Manual  
000399-001 B**

We appreciate your comments regarding any problems or suggestions related to AST Research products. Please use this form to communicate any observations that you have concerning the improvement of either the product itself or the product documentation provided in this manual.

**Submitter Information**

Submitter's name:

Address:

**Product/Manual Comments and Suggestions**

Please mail this form to:

AST Research, Inc.  
Attn: Product Marketing  
2121 Alton Ave.  
Irvine, CA 92714-4992



Index





## INDEX

---

<Ctrl>-<Break>, 2-8, 2-14, 3-15, 4-8, 8-2  
<Ctrl>-<C>, 2-8, 2-15, 3-15, 4-8, 8-2  
<Ctrl>-<H>, 4-8  
<Ctrl>-<J>, 4-8  
<Ctrl>-<N>, 4-8  
<Ctrl>-<P>, 4-9  
<Ctrl>-<S>, 2-8, 4-9  
<Ctrl>-<X>, 4-9  
<Ctrl>-<Z>, 2-18, 2-19

### A

Adding lines (EDLIN), 5-16  
Align type, 6-33  
ANSI.SYS, 8-7  
APPEND, 5-16  
Archive attribute, 3-10, 3-11  
Arguments, 2-6  
ASSIGN, 3-8, 3-9  
Asterisk (\*)  
    as system prompt in EDLIN, 5-2, 5-3  
    as wildcard symbol, 1-10, 1-11  
ATTRIB, 3-10, 3-11  
AUTOEXEC.BAT file, 2-16 - 2-18

### B

Background printing, 3-101 - 3-104  
BACKUP, 3-12 - 3-14, 3-114  
BAT extension, 2-4, 2-6, 2-13  
Batch processing, 2-13 - 2-15  
Brackets, 3-1  
BREAK, 3-15, 8-2  
BUFFERS, 8-3

## C

Changing directories, 3-16, 3-17

CD, see CHDIR

CHDIR (change directory), 3-16, 3-17

CHKDSK, 3-18 - 3-24

CLS (clear screen), 3-25

Color adapter, 3-82 - 3-84

Combine types, see LINK

Combining segments, see LINK

COMMAND, 3-26, 3-27

COMMAND.COM (command processor), 3-26, 3-27, 3-61, 3-127,  
3-128, 8-17

Command format, 3-1, 3-2

Command lines, (LINK), 6-5 - 6-7

Commands

arguments, 2-6

ASSIGN, 3-8, 3-9

ATTRIB, 3-10, 3-11

BACKUP, 3-12 - 3-14, 3-114

BREAK, 3-15, 8-2

CD, see CHDIR

CHDIR (change directory), 1-25, 3-16

CHKDSK, 1-5, 3-18 - 3-24

CLS (clear screen), 3-25

COMMAND, 3-26, 3-27

COMP (compare), 3-28 - 3-31

COPY, 3-32 - 3-39

CTTY, 3-40

DATE, 3-41, 3-42

DEL, 3-43

DIR (directory), 1-4, 3-44, 3-45

DISKCOMP (disk compare), 3-46 - 3-51

DISKCOPY, 3-52 - 3-57

ECHO, 3-142  
EXE2BIN (executable to binary), 3-58 - 3-60  
EXIT, 3-61  
External, 2-4  
FDISK, 3-62 - 3-74  
FIND, 3-75 - 3-77  
FOR, 3-143 - 3-144  
FORMAT, 3-78 - 3-81  
GOTO, 3-145  
GRAFTABL, 3-82  
GRAPHICS, 3-83, 3-84  
IF, 3-146, 3-147  
Internal, 2-3  
JOIN, 3-85, 3-86  
KEYBxx, 3-87 - 3-88  
LABEL, 3-89, 3-90  
MKDIR (make directory), 3-91, 3-92  
MODE, 3-93 - 3-97  
MORE, 3-98  
Option, 2-6  
Overview, 2-2 - 2-8  
PATH, 3-99, 3-100  
PAUSE, 3-148  
PRINT, 3-101 - 3-104  
PROMPT, 3-105 - 3-107  
RECOVER, 3-108, 3-109  
REM, 3-149  
REN (rename), 3-110, 3-111  
REPLACE, 3-112, 3-113  
RESTORE, 3-114, 3-115  
RMDIR (remove directory), 1-26, 3-116  
SELECT, 3-117, 3-118

- SET, 3-119, 3-120
- SHARE, 3-121, 3-122
- SHIFT, 3-150
- SORT, 3-123, 3-124
- SUBST, 3-125, 3-126
- SYS, 3-127, 3-128
- TIME, 3-129, 3-130
- TREE, 3-131, 3-132
- TYPE, 3-133
- VER (version), 3-134
- VERIFY, 3-135
- VOL, 3-136
- XCOPY, 3-137
- Compilers, 6-6, 6-22, 6-24, 6-30
- CONFIG.SYS, 8-1 - 8-17
- COPY, 3-32 - 3-39
- Copying
  - disks, 3-52 - 3-57
  - files, 3-32 - 3-39
  - lines (EDLIN), 5-4 - 5-7, 5-17 - 5-19
- COUNTRY, 8-4, 8-5
- Country code, 3-118
- CTTY, 3-40

## D

- DATE, 3-41, 3-42
- Date format, 3-117, 3-118, 8-4, 8-5

## DEBUG

## commands

- ASSEMBLE, 7-8 - 7-10
- COMPARE, 7-11
- DUMP, 7-12
- ENTER, 7-14, 7-15
- FILL, 7-16
- GO, 7-17, 7-18
- HEX, 7-19
- INPUT, 7-20
- LOAD, 7-21, 7-22
- MOVE, 7-23
- NAME, 7-24 - 7-26
- OUTPUT, 7-27
- QUIT, 7-28
- REGISTER, 7-29 - 7-32
- SEARCH, 7-33
- TRACE, 7-34, 7-35
- UNASSEMBLE, 7-36, 7-37
- WRITE, 7-38

## errors, 7-40

DEL, 3-43

## Deleting

- directories, 1-26, 3-16
- files, 3-43
- in EDLIN, 5-20 - 5-25

Destination drive, 2-8

## Device

- In CONFIG.SYS, 8-1, 8-6 - 8-13
- names, 1-6
- operation modes, 3-93 - 3-97

DIR (directory), 1-4, 3-44, 3-45

## Directories

- changing, 1-25, 3-44, 3-45
- copying, 3-137 - 3-140
- creating, 1-25, 3-91
- definition, 1-3
- deleting, 1-26
- displaying, 1-23, 1-24, 3-44, 3-45
- making, 1-25, 3-91
- multilevel, 1-13, 1-14
- overview, 1-12, 1-13
- removing, 1-26, 3-116
- root, 1-14
- working, 1-14

Disk swapping, 3-148, 6-18

DISKCOMP (disk compare), 3-46 - 3-51

DISKCOPY, 3-52 - 3-57

## Disks

- backing up, 3-12 - 3-14
- checking, 1-5, 3-18 - 3-24
- comparing, 3-46 - 3-51
- formatting, 3-78 - 3-81

## Displaying

- directories, 1-23, 1-24, 3-44, 3-45
- files, 3-133

## Drive

- letters, 3-8, 3-9
- name, 1-7, 2-6

DRIVER.SYS, 8-8 - 8-10

Dummy parameters, 2-19 - 2-21

## E

ECHO, 3-142

## Editing

files, 5-1

keys, 4-1 - 4-7, 5-4 - 5-14

text (EDLIN), 5-24, 5-25

Editing and function keys, 4-1 - 4-9

Editing commands, see EDLIN

## EDLIN

## commands

APPEND, 5-16

COPY, 5-17 - 5-19

DELETE, 5-20 - 5-23

EDIT, 5-24, 5-25

END, 5-26

INSERT, 5-27 - 5-31

LIST, 5-32 - 5-34

MOVE, 5-35 - 5-37

PAGE, 5-38

QUIT, 5-39

REPLACE, 5-40 - 5-43

SEARCH, 5-44 - 5-46

TRANSFER, 5-47

WRITE, 5-48

copy characters, 5-5, 5-6

copy template, 5-7

ending the edit session, 5-3, 5-26

insert mode, 5-11, 5-12

new template, 5-13, 5-14

skip, 5-8, 5-9

starting the edit session, 5-2

overview, 5-1

Ellipsis, 3-1

Environment variables, 6-10

ERASE, see DEL



EXE files, 7-22

EXE2BIN (executable to binary), 3-58 - 3-60

Executable files

- converting, 3-58 - 3-60

- creating, 6-1

- definition, 2-4

Executable image, 6-32

EXIT, 3-61

Extensions, 1-6, 2-4

External commands, 1-21, 2-4

## F

FCBS, 8-14

FDISK, 3-62 - 3-74

File

- allocation table, 1-3

- definition, 1-2

- handles, 8-15

- header, 6-16

- system, 1-15 - 1-17

- usage

  - appending, 3-32 - 3-39

  - combining, 3-32 - 3-39

  - comparing, 3-28 - 3-30

  - copying, 3-32 - 3-39, 3-137 - 3-140

  - deleting, 3-43

  - displaying, 3-133

  - editing, 5-1 - 5-48

  - in CONFIG.SYS, 8-15

  - locking, 3-121, 3-122

  - naming, 1-5, 1-8, 2-6, 3-110, 3-111

  - printing, 3-101 - 3-104

  - protecting, 1-11

  - recovering, 3-108, 3-109

  - renaming, 3-110, 3-111

  - restoring, 3-114, 3-115

  - sharing, 3-121, 3-122

  - sorting, 3-123, 3-124

  - verifying, 3-135

Filename extensions, 1-6, 2-4  
FILES (CONFIG.SYS), 8-15  
Filters, 2-11  
FIND, 3-75 - 3-77  
Fixups (LINK), 6-36, 6-37  
Flags, 7-30  
FOR, 3-143, 3-144  
FORMAT, 1-4, 3-78 - 3-81  
Frame number, canonical, 6-34

## G

GOTO, 3-145  
GRAFTABL (graphics table), 3-82  
GRAPHICS, 3-83, 3-84  
Graphics adapter, 3-82 - 3-84  
Groups (LINK), 6-36  
GW-BASIC, 2-18

## H

Hard disk, 3-62 - 3-74, 3-79, 3-108, 3-109  
Hidden files, 1-5, 3-11  
High start address, setting, 6-27

## I

IF, 3-146 - 3-147  
Illegal filenames, 1-8  
Input, 2-9  
Inserting text (EDLIN), 5-27 - 5-31  
Internal commands, 1-22, 2-3

## J

JOIN, 3-85, 3-86

## K

KEYBxx, 3-87 - 3-88

Keyboard, 3-87 - 3-88, 3-117, 3-118

## L

LABEL, 3-89, 3-90

LAST DRIVE, 8-16

Lib, environment variable, 6-10

Library files, 6-5 - 6-11

### LINK

- align type, 6-35

- groups, 6-38

- library files, 6-5 - 6-11

- library search, 6-25

- map file, 6-12, 6-13

- options

  - /cparmaxalloc, 6-27

  - /dosseg, 6-34

  - /dsallocate, 6-30

  - /exepack, 6-21

  - /help, 6-18

  - /high, 6-29

  - //lnumbers, 6-23

  - /map, 6-22

  - /nodefaultlibrarysearch, 6-25

  - /nogroupassociation, 6-31

  - /noignorecase, 6-24

  - /overlayinterrupt, 6-32

  - /pause, 6-19

  - /segments, 6-33

- overlays, 6-32
- overview, 6-1
- preserving case-sensitivity, 6-24
- with command line, 6-5 - 6-7
- with prompts, 6-2, 6-3
- with response file, 6-8 - 6-10
- search paths, 6-10, 6-11
- temporary file, 6-14
- using, 6-2 - 6-4

Listing lines (EDLIN), 5-32 - 5-34

## M

Map files, see LINK

Merging files (EDLIN), 5-47

MKDIR (make directory), 3-91

MODE, 3-93 - 3-97

MORE, 3-98

Moving lines, 5-35 - 5-37

## N

Networks, 3-79, 3-121, 3-122, 8-16

## O

Options, command, 2-5, 2-6

Output, 2-9, 2-10

Overlays (LINK), 6-32

## P

PAGE, 5-38  
Parameters, 2-19 - 2-21  
Parent directory, 1-20, 3-16  
PATH, 3-99, 3-100  
Paths, 1-18 - 1-22, 2-6, 3-131, 3-132  
PAUSE, 3-148  
Piping, 2-11, 2-12  
PRINT, 3-101 - 3-104  
Printers, 3-93 - 3-97, 3-101 - 3-104  
PROMPT, 3-105 - 3 - 107  
Protecting files, 1-11

## Q

Question mark (?), 1-9  
Quitting (EDLIN), 5-3

## R

Read-only files, 3-10, 3-11, 3-43  
RECOVER, 3-108, 3-109  
REM, 3-149  
REN (rename), 3-110, 3-111  
REPLACE, 3-112, 3-113  
Replacing text (EDLIN), 5-40 - 5-43  
Reserved filenames, 1-8  
RESTORE, 3-13, 3-114, 3-115  
RMDIR (remove directory), 1-26, 3-116  
Root directory, 1-4, 1-13 - 1-17

**S**

Search paths, see also LINK, 3-99, 3-100  
Searching for text, see also EDLIN, 3-75 - 3-77  
Segment number, see LINK  
Segment order, see LINK  
SELECT, 3-117, 3-118  
SET, 3-119, 3-120, 6-10  
SHELL, 8-17  
SHARE, 3-121, 3-122  
SHIFT, 3-150  
SORT, 2-11, 3-123, 3-124  
Source drives, 2-8  
Special characters, 1-7  
Special editing keys, 5-4 - 5-14  
Starting EDLIN, 5-2, 5-3  
Stop activity, 2-8, 3-15  
String, 3-2  
Subdirectory, 1-12 - 1-14, 3-91  
SUBST, 3-125, 3-126  
Switches, 2-6  
SYS, 3-127, 3-128  
System files, 3-11, 3-43, 3-127, 3-128

**T**

Target drives, 2-8  
TIME, 2-129  
Transferring system files, 3-127, 3-128  
TREE, 3-131, 3-132  
TYPE, 3-133

## V

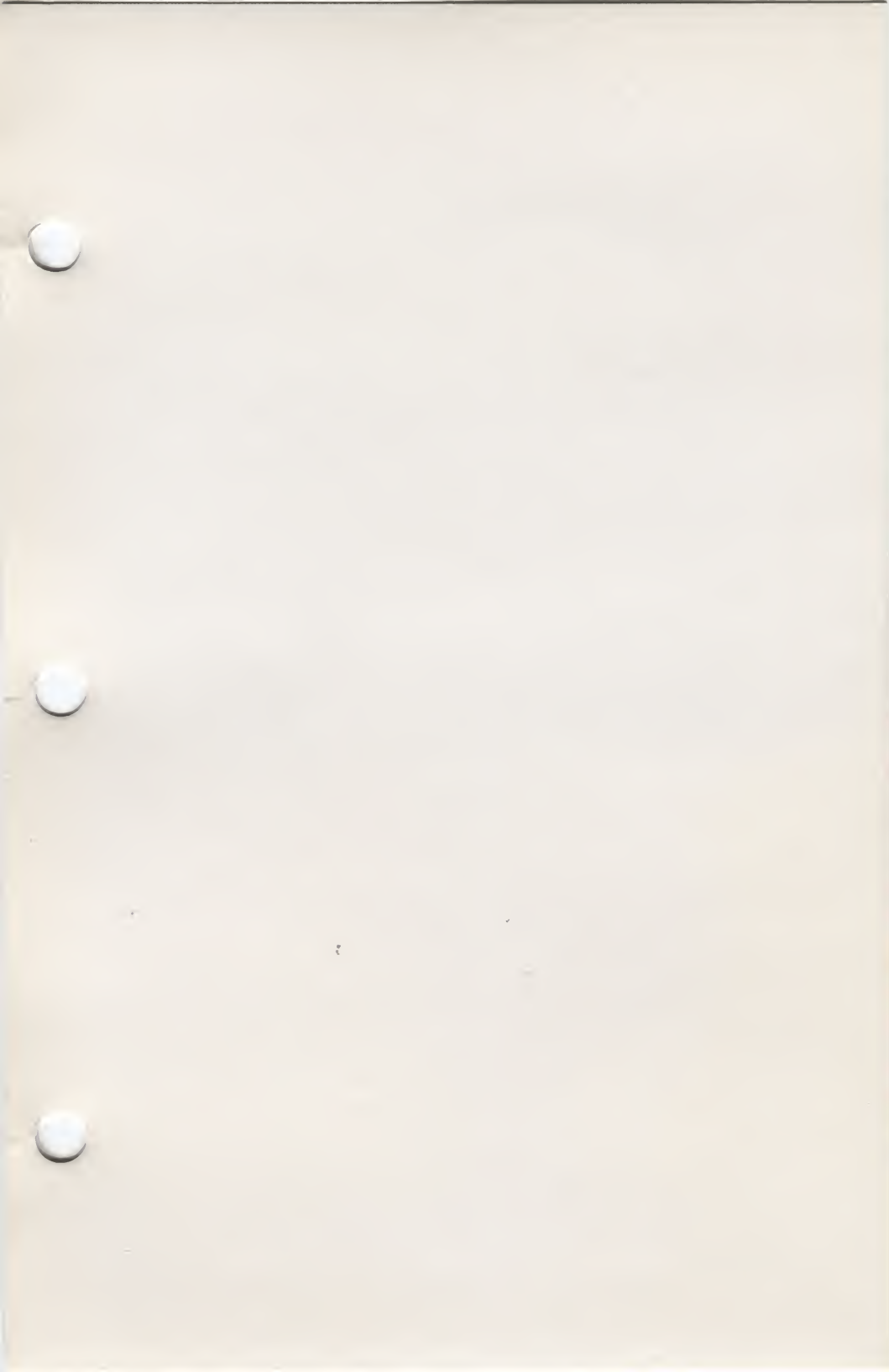
VDISK.SYS, 8-11 - 8-13, 8-16  
VER (version), 3-134  
VERIFY, 3-135  
Version number, displaying, 3-134  
Virtual drive, 3-125, 3-126, 8-11 - 8-13  
VM.TMP (LINK), 6-14  
VOL, 3-136  
Volume ID, 1-24, 3-78, 3-79, 3-81, 3-136  
Volume label, see Volume ID

## W

Wildcards, 1-9, 1-10  
Working directory  
    creating, 1-25  
    definition, 1-14  
    displaying, 1-23, 1-25  
    shorthand notation, 1-20  
WRITE (EDLIN), 5-48

## X

XENIX, 3-63  
XCOPY, 3-137





000399-001 B





MS-DOS 3.2

REL. 1.2

AST.  
*Premium*  
COMPUTER PRODUCTS

© Copyright 1987 Licensed Material-Program Property of AST Research, Inc., 2121 Alton Avenue, Irvine, CA 92714-4992.  
© Copyright 1981, 1987, Microsoft Corporation. © Copyright 1984, 1987, PHOENIX Technologies, Ltd.

MS-DOS Supplement

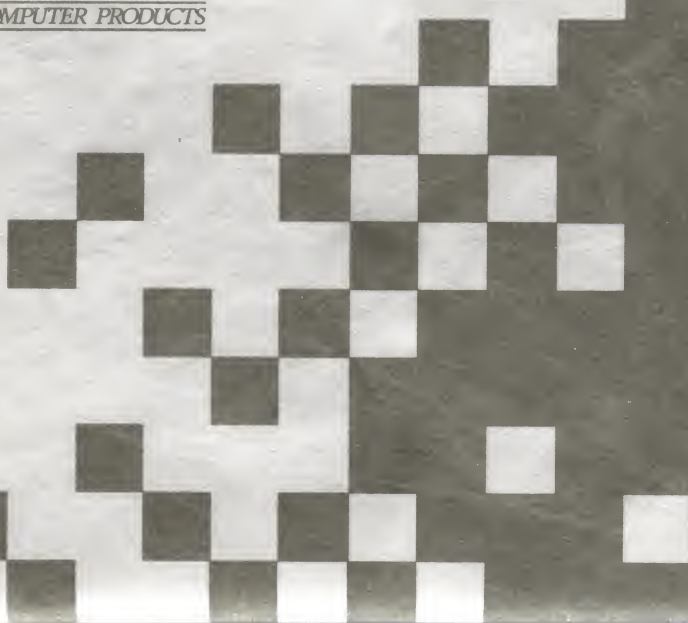
Release 1.2

AST.  
*Premium*  
COMPUTER PRODUCTS

© Copyright 1987 Licensed Material - Program Property of AST Research, Inc., 2121 Alton Avenue, Irvine, CA 92714-4992.  
© Copyright 1981, 1987, Microsoft Corporation. © Copyright 1984, 1987, PHOENIX Technologies, Ltd.  
Unauthorized use or distribution is strictly prohibited by federal law.

AST.  
*Premium*  
COMPUTER PRODUCTS

AST  
RESEARCH INC.



**For extended media life—  
here's how to take care of your flexible disk**



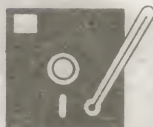
Protect  
Proteger  
Protéger  
Schützen  
保護



Never  
Nunca  
Jamais  
Nie  
絶対禁止



No  
No  
Non  
Nein  
注意



10°C—52°C  
50°F—125°F



Insert Carefully  
Insertar  
Insérer avec soin  
Sorgfältig Einsetzen  
插入注意



Never  
Nunca  
Jamais  
Nie  
絶対禁止



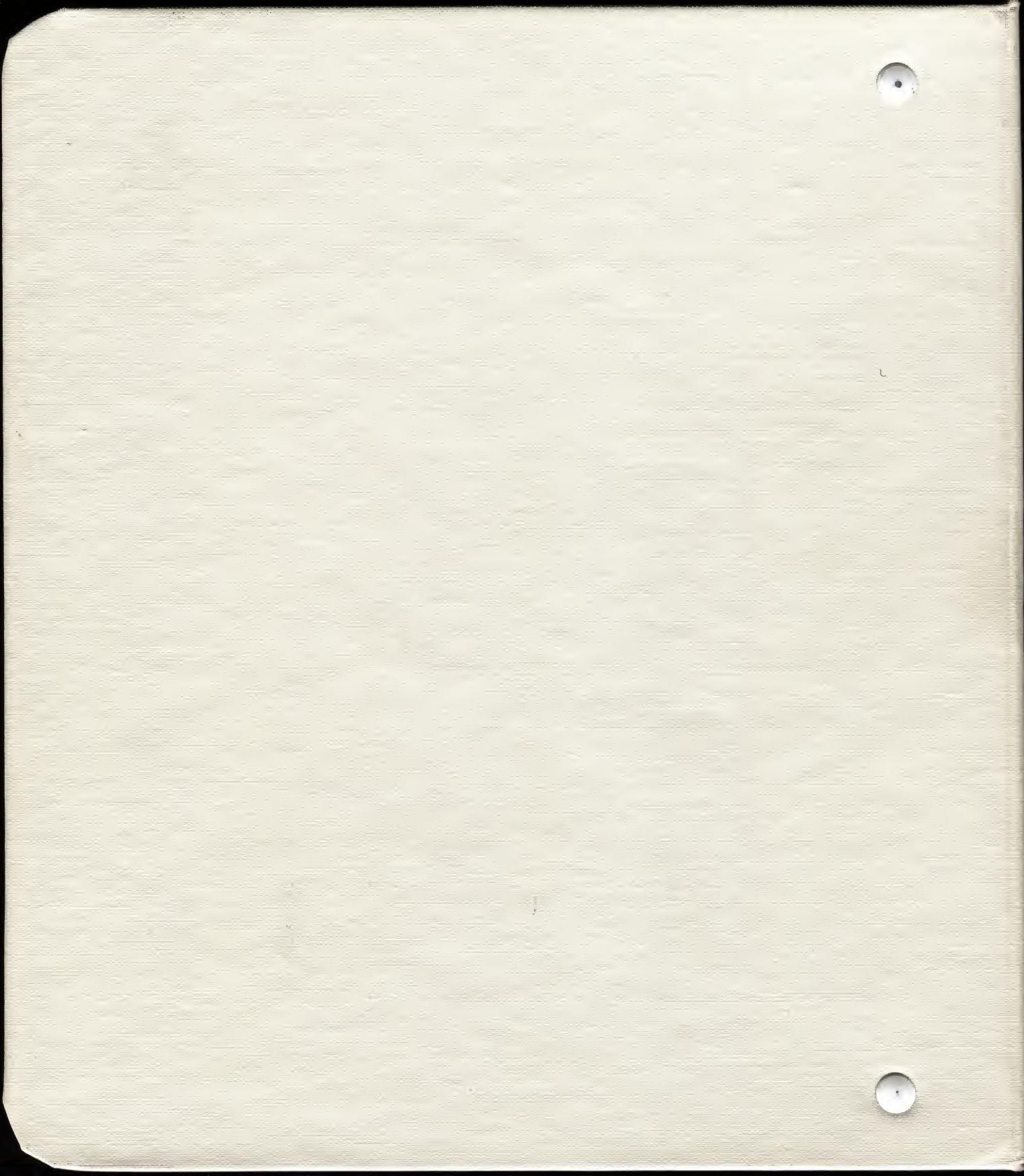
Insert Carefully  
Insertar  
Insérer avec soin  
Sorgfältig Einsetzen  
插入注意



Never  
Nunca  
Jamais  
Nie  
絶対禁止







**AST**  
RESEARCH INC.

# *Operating System*

—AST—  
*Premium*  
COMPUTER PRODUCTS